

INPUT
Publicación práctica para usuarios de
commodore
Revista mensual 1986
Precio 350 Ptas
Año 2 Número 14
MANEJO DE TEXTOS
CARACTERES PARA JUEGOS
COMO SE ALMACENAN LOS NUMEROS
GRAFICOS PAGINADOS
commodore 64

SI BUSCAS LO MEJOR ERBE Software LO TIENE

ALLEYKAT

SEGURO QUE CON TU ORDENADOR HAS PARTICIPADO EN CARRERAS DE COCHES Y MOTOS, PERO NUNCA LO HAS HECHO EN UNA COMPETICION DE NAVES MONOPLAZAS DEL SIGLO XXI EN LA MAS ALUCINANTE CARRERA ESPACIAL QUE HAYAS VISTO NUNCA.



DISTRIBUIDOR EXCLUSIVO PARA ESPAÑA: ERBE SOFTWARE. C/. STA. ENGRACIA, 17
28010 MADRID. TEL. (91) 447 34 10 - DELEGACION BARCELONA, AVDA. MISTRAL, N.º 10 - TEL. (93) 432 07 31



AÑO 2 NUMERO 14

DIRECTOR:
Alejandro Diges
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Luis R. Palencia, Christophe Païs, Francisco Tórtola, Benito Román, Esther de la Cal.
INPUT Commodore es una publicación de PLANETA-DE AGOSTINI, S. A.
GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: Grupo Jota
Madrid: c/ General Varela, 35
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.°
Teléf. 250 23 99
FOTOMECANICA: Ochoa, S. A.
COMPOSICION: EFCA, S. A.
IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: M. 27.884-1985
SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12
REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13
DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.
El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

INPUT Commodore es una publicación controlada por



INPUT commodore

SUMARIO

LA CRISIS DE LOS Ks

A estas alturas del año, cuando sólo nos resta por ver lo que pueda ofrecernos el **SIMO,** y ya se intuye la presión publicitaria de las campañas de Navidad, es el momento de hacer un repaso de lo que microinformáticamente ha sido el año 86. Lo iniciamos, ilusionadamente, con la llegada de los 128 K. Tanto **Commodore** como **Sinclair** parecían dispuestos a satisfacer la demanda de Ks de memoria que exigían los usuarios para sentir que sus máquinas no se quedaban atrás en el imparable avance de los micros por alcanzar las capacidades mitológicas de los grandes ordenadores. Sin embargo, en estos últimos meses, los fabricantes han decidido reconsiderar el camino que habrían de tomar sus nuevos productos.

Los que intuían una progresión geométrica de las capacidades de memoria que se proyectaba así hacia el infinito, se han encontrado con que **Commodore** ha diseñado una bonita carcasa para su clásico **C-64.** Y que **Sinclair-Amstrad,** aunque ha hecho bastante más por su **128** (le ha incorporado un cassette, un *chip* de sonido y alguna cosa más) no ha superado el listón de los 128 K alcanzado el invierno pasado ni en un solo bit. Si no hubiera sido por el *software* el año del Cometa, del **Amiga** y de la caída de los precios de los **PC** compatibles, habría sido un año gris para nuestros micros. Han sido los creadores de programas los que han dado el do de pecho y han puesto en evidencia las auténticas capacidades de estas magníficas máquinas. De su trabajo tenéis una amplia muestra en la revista de *software* de este número. Disfrutad con ellos de vuestro micro, y el año que viene ya veremos...

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** P.º de la Castellana, 93. Planta 14. 28046 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido ____________________

Segundo título elegido ____________________

Tercer título elegido ____________________

Programa que te gustaría conseguir ____________________

Qué ordenador tienes ____________________

Nombre ____________________

1.er Apellido ____________________

2.º Apellido ____________________

Fecha de nacimiento ____________________

Teléfono ____________________

Dirección ____________________

Localidad ____________________

Provincia ____________________

GRAFICOS PAGINADOS

Conocer el fundamento de los gráficos paginados abre un mundo hacia las posibilidades de esta técnica. Algún ejemplo demuestra como funciona en el Commodore 64.

Los gráficos paginados —técnica de conmutar desde una página gráfica a otra— ofrecen un potencial considerable en muchos tipos diferentes de aplicaciones en las que se precisa un cambio rápido de los datos de una pantalla a los de otra. Aunque su utilización obvia es la animación de imágenes con ordenador, pueden ser aplicadas en usos más serios, siendo un ejemplo los diagramas que muestran cifras en pantalla, tal como es el caso de programas financieros.

El principio que se esconde tras los gráficos paginados consiste en definir y después confinar en la memoria los datos correspondientes a pantallas diferentes. Estos datos pueden tomar la forma de gráficos en alta o baja resolución o incluso texto, tal vez una combinación de ellos. Cada una de estas pantallas puede ser llamada en una sucesión muy rápida, sin necesidad de tiempo para calcular cómo llenar cada una de las pantallas.

El tiempo para llenar la pantalla sigue siendo un requerimiento, pero sólamente es necesario hacerlo una vez para cada pantalla diferente antes de comenzar a visualizarlas. Después se confinan en zonas adecuadas de la memoria, pudiendo reclamarse instantáneamente cuando se requieran.

RESTRICCIONES DE LA MEMORIA

Cada «página» de datos de pantalla requiere una determinada cantidad de memoria. La cantidad necesaria varía debido a que cuantos más colores utilices, y mayor sea la resolución de los gráficos más se necesitará. En algunos



casos existen severas restricciones de memoria en función del programa. Es entonces cuando hay que reducir el uso de la pantalla a sólamente una fracción de su tamaño normal. También es necesario a menudo sacrificar colores y resolución. Es obligado dejar espacio en la RAM para el programa.

La técnica de la paginación puede llamar a pantallas individuales depositadas en la memoria, siguiendo cualquier orden y más de una vez en cualquier secuencia que sea preciso.

Por tanto es factible construir una secuencia de paginación de quizás ocho pantallas, aunque en realidad existan muchas menos presentes en la memoria. Esta útil técnica de ahorro de memoria es especialmente efectiva si tenemos cuidado de asegurar que los gráficos de las pantallas repetidas intermediamente no hagan nada en detrimento del «flujo» de la animación.

Por ejemplo, un programa que muestre a un hombre de alambre andando, podría utilizar las páginas 1,2,3,4,5,1,2,3,4,5, etc. Sin embargo donde el espacio sea una limitación, la similitud entre dos imágenes, p. ej. las 2 y 4 así como las 3 y 5, es tal que poco se nota si ambas son en realidad la misma imagen. Esto produce una secuencia 1,2,3,2,3,1, etc. La secuencia sigue siendo de cinco imágenes, pero sólo se precisan tres pantallas.

En el **Commodore 64** sólamente se pueden almacenar dos pantallas de



alta resolución simultáneamente. Contener una imagen de una pantalla por mapa de bits significa 8 Kbytes de memoria. Afortunadamente se pueden utilizar secciones más pequeñas de la pantalla de cara a incrementar el número de páginas disponibles para este tipo de gráficos y esto es lo que hace el ejemplo que sigue para obtener un total de cinco páginas.

Este programa hace uso del cartucho **Simon's BASIC**. Este cartucho ocupa la RAM situada entre las direcciones 8192 y 16384, siendo conveniente utilizar la RAM que hay más arriba para las páginas gráficas.

Teclea para Commodore-64

```
20 POKE 51,255:POKE 52,29:
   POKE 55,255:POKE 56,29:
   CLR
```

```
30 GOSUB 220
40 D=64
50 FOR N=0 TO 4
60 HIRES 0,1:MULTI 7,4,3:
   COLOUR 6,0
70 FOR Z=1 TO 12:FOR ZZ=1 TO
   3:LINE Z*Z+ZZ*(N+1),0,Z*Z
   +ZZ*(N+1),100,ZZ
75 PLOT RND(1)*160,RND(1)*
   100,RND(1)*3+1:NEXT ZZ,Z
80 FOR ZZ=1 TO 3:LINE 0,(N*
   13)+ZZ*(N+1),159,(N*19)+
   ZZ*(N+1),ZZ:NEXT ZZ
90 FOR ZZ=1 TO 3:CIRCLE 10+
   N*35,60,20-ZZ*3,10+N*3,ZZ
   :NEXT ZZ
100 GOSUB 430:D=D+12
110 NEXT N
130 BLOCK 0,0,159,199,3
140 TEXT 0,100,
    "ANIMACION",0
    ,8,19
150 D=64:FOR N=0 TO 4
170 GOSUB 440:D=D+12:FOR T=1
    TO 15:NEXT T
190 NEXT N
200 GOTO 150
220 FOR Z=7680 TO 7738:READ
    X:POKE Z,X:NEXT Z:RETURN
230 DATA 169,0,141,14,220,
    169,53,133,1
240 DATA 169,0,133,251,133,
    253,169,224,133,252,169,
    64,133,254,160,0
250 DATA 177,251,145,253,192
    ,63,208,16,165,252,201,
    235,208,10
260 DATA 162,1,142,14,220,
    162,55,134,1,96,200
270 DATA 208,229,230,252,230
    ,254,76,25,30
430 POKE 7700,D:POKE 7706,
    251:POKE 7708,253:SYS
    7680:RETURN
440 POKE 7700,D:POKE 7706,
    253:POKE 7708,251:SYS
     7680:RETURN
```

```
DESENSAMBLADO DEL CODIGO
MAQUINA.
========================
,1E00 A9 00     LDA #$00
,1E02 8D 0E DC  STA $DC0E
,1E05 A9 35     LDA #$35
,1E07 85 01     STA $01
,1E09 A9 00     LDA #$00
,1E0B 85 FB     STA $FB
,1E0D 85 FD     STA $FD
,1E0F A9 E0     LDA #$E0
,1E11 85 FC     STA $FC
,1E13 A9 40     LDA #$40
,1E15 85 FE     STA $FE
,1E17 A0 00     LDY #$00
,1E19 B1 FB     LDA ($FB),Y
,1E1B 91 FD     STA ($FD),Y
,1E1D C0 3F     CPY #$3F
,1E1F D0 10     BNE $1E31
,1E21 A5 FC     LDA $FC
,1E23 C9 EB     CMP #$EB
,1E25 D0 0A     BNE $1E31
,1E27 A2 01     LDX #$01
,1E29 8E 0E DC  STX $DC0E
,1E2C A2 37     LDX #$37
,1E2E 86 01     STX $01
,1E30 60        RTS
,1E31 C8        INY
,1E32 D0 E5     BNE $1E19
,1E34 E6 FC     INC $FC
,1E36 E6 FE     INC $FE
,1E38 4C 19 1E  JMP $1E19
```

Este programa comienza ajustando el tope para el texto en BASIC suficientemente debajo del **Simon's** para aceptar (ver el CLR) una corta rutina en código maquina que maneje la paginación entre pantallas. Esta rutina se carga en la memoria por medio de la subrutina que comienza en la línea 220, siendo accedida posteriormente, tanto para almacenamiento como para llamadas, por sentencias SYS.

GANADORES DE LOS MEJORES DE INPUT COMMODORE

En el sorteo correspondiente al número 14 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Joan Minguell Arrufat	Borges Blanques (Lérida)	Skyfox
Alfredo Rodríguez Nogeira	Valladolid	Rambo
Pedro Castaño Pérez	H. de Llobregat (Barcelona)	Skyfox
Manuel Martín Giménez	Almería	Saboteur
Carlos Mateo Ballorca	Burgos	Kung Fu Master
L. Carlos Martínez Zapata	Madrid	Green Beret
Rafael Pérez Delgado	Huelva	Green Beret
J. Javier Muñoz Romero	Barcelona	Pistop II
J. Ramón Rogada Arribas	Mieres (Asturias)	Green Beret
Antonio Martínez Alvarez	Barcelona	«V»

El primero de los punteros de la pantalla es establecido en la línea 40 en el comienzo del área de RAM disponible debajo del **Simon's**. El programa continúa con una rutina de dibujo para las cinco pantallas, recurriendo al bucle FOR...NEXT de la línea 50.

Las líneas 60 a 90 manejan el diseño en sí. Cuando está completo el trabajo de dibujo, se accede a la rutina con POKEs de la línea 440, haciendo una llamada SYS a la rutina en código máquina. Así es confinado el primer bloque de dibujo en su lugar adecuado de la memoria.

Después continúa el bucle trazando las otras cuatro pantallas, situadas cada una en un bloque de 3 Kbytes por medio del ajuste de punteros realizado con D = D+12 en la línea 100.

La línea 130 limpia la pantalla y la tiñe de azul claro, imprimiendo con la línea 140 un mensaje de comienzo de la secuencia que le sigue.

El puntero de memoria D es inicializado a su valor original de 64, la dirección inicial de la primera pantalla que existe inmediatamente debajo del final del **Simon's**.

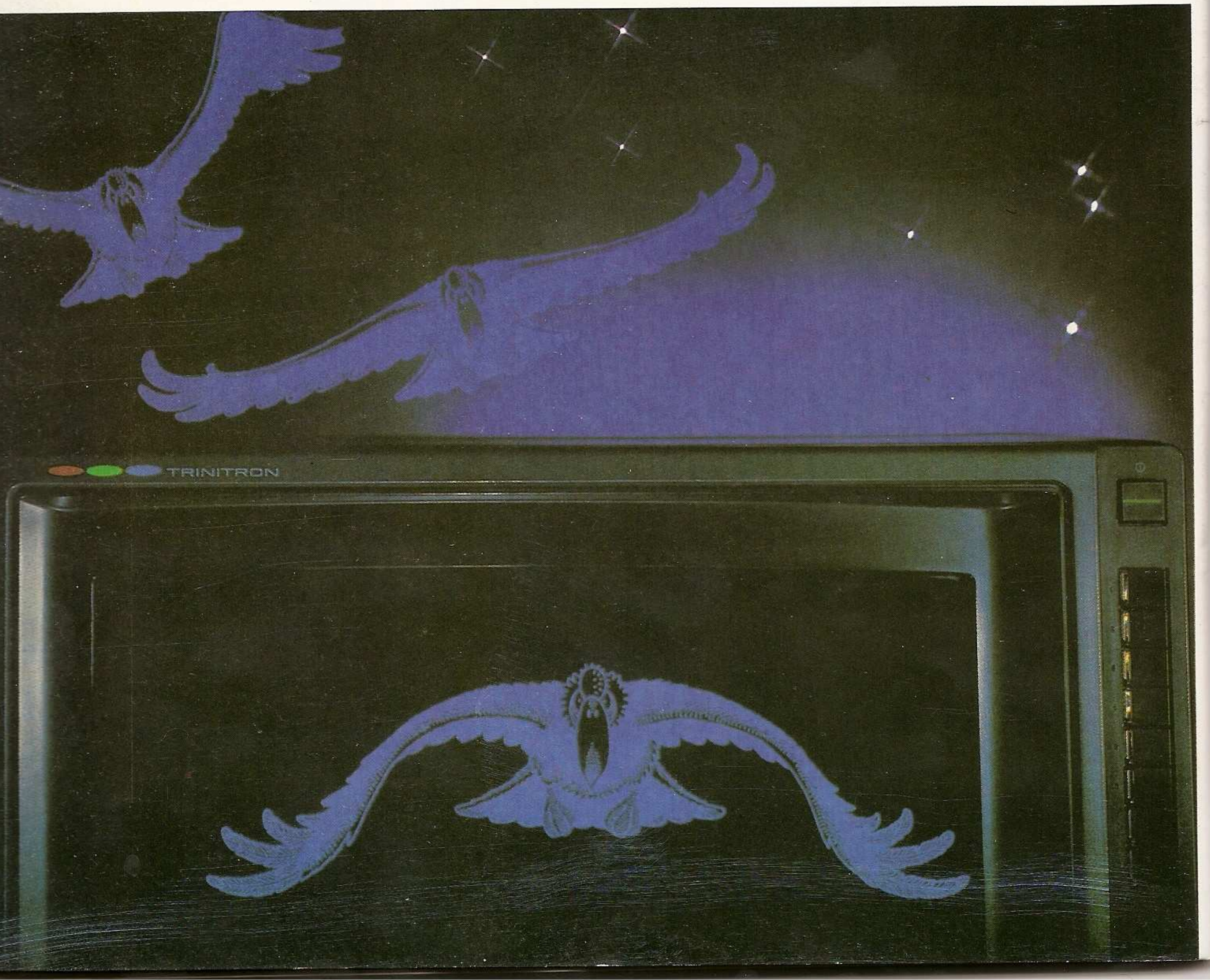
La primera de las cinco pantallas es llamada a partir de este momento con la línea 440, que vuelve a acceder a la rutina en código máquina de la dirección 7680.

El puntero es ajustado en la línea 170 (D = D+12) para la siguiente pantalla. Se produce una pequeña demora de tiempo con el bucle T=1 TO 15, y la nueva pantalla pasa a solaparse sobre la anterior. Se repite lo mismo a través de los cinco cuadros, restaurándose el puntero original al completarse el bucle.

Puedes utilizar la misma rutina en código máquina para tus propios programas, acordándote de establecer la coordenada de la esquina inferior derecha de la pantalla y ajustar el valor del byte bajo en lugar del 63 y el valor del byte alto sustituyendo al 253 de la línea 250 cuando utilices el **Simon's BASIC**.

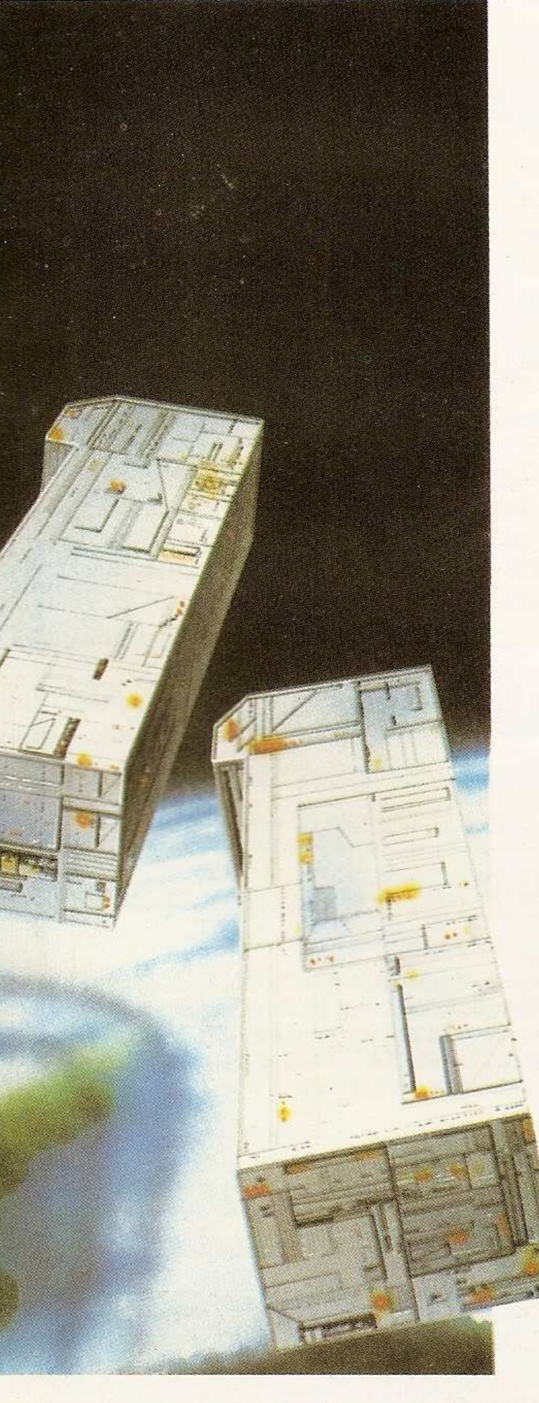
Si no utilizas el cartucho para trabajar en alta resolución, recuerda que las direcciones de la memoria de pantalla comienzan en 8192 (bajo 0, alto 32) y no en 57344 como con el **Simon's**.

Si tu diseño implica un tamaño de pantalla inferior, libera más memoria para paginación, ajusta los bucles FOR...NEXT, incrementando el valor del puntero D para adecuarse al número y tamaño de la memoria de cada página.



- EXPONENTES
- COMA FLOTANTE
- NUMEROS NEGATIVOS
- ALMACENAMIENTO DE NUMEROS EN MEMORIA

```
94 NEXT F:F=0
96 IF F=LEN(N$) THEN N$=
   LEFT$(N$,LEN(N$)-1):
   GOTO 100
98 IF F<>0 THEN N$=LEFT$
   (N$,N$,F-1)+MID$(N$,F+1)
   +"."+MID$(N$,F+2):GOTO 80
100 N$=N$+"0":GOTO 80
110 IF RIGHT$(N$,1)="."THEN
    N$=LEFT$(N$,LEN(N$)-1)
120 GOTO 180
130 IF ABS(N)>=1 THEN 170
140 N=N*10
142 FOR F=1 TO LEN(N$):IF
    MID$(N$,F,1)="."THEN 146
144 NEXT F:F=0
146 IF F=1 THEN N$="0"+MID$
    (N$,,2):GOTO 130
150 IF F<>0 THEN N$=LEFT$(N$
    ,F-2)+"."+MID$(N$,F-1,1)
    +MID$(N$,F+1):GOTO 130
160 N$="."+N$:GOTO 130
170 IF VAL(A$)<0 THEN N$="-"
    +N$
180 PRINT"[SHIFT+CLR/HOME]";
    A$;"[CRSR dcha.]IGUAL A"
    :PRINT N$
190 GOTO 20
200 IF ABS(N)<1 THEN 220
210 IF ABS(N)>=10 THEN E=E+1
    :N=N/10:GOTO 210
215 GOTO 230
220 IF ABS(N)<=1 THEN E=E-1:
    N=N*10:GOTO 220
230 PRINT"[SHIFT+CLR/HOME]
    [CRSR dcha.]";A$;"[CRSR
    dcha.]IGUAL A":PRINT N;
    "[CRSR izqda.]";:IF E<>0
    THEN PRINT"E";E
240 PRINT:GOTO 20
```



Cuando ejecutes este programa, el ordenador espera que escribas un número. Deberás teclear un número positivo en forma clásica o exponencial. No te preocupes si continuan sin saber lo que es un exponencial, pronto lo harás. Para convertir un número escrito con exponente en uno más significativo para ti, necesitas simplemente multiplicar los primeros dígitos del número (conocidos por mantisa) por 10 elevado a la potencia de la cantidad que sigue a E, que es conocido como exponente de ese número.

Aunque esto suena bastante complicado, si lo trabajas paso a paso, rápidamente se ve claro. Toma 1.23E4 para convertirlo a la forma usual, multiplica la mantisa (1.23) por 10 elevado a la potencia del exponente, que es igual a 1,23*10.000. Así que 1,23E4 es lo mismo que 12,300.

Para cualquier número, la mantisa toma siempre un valor de un dígito seguido por una coma decimal, variando por tanto entre 1,0 y 9,999999...

Los números negativos se manejan igualmente, sólo que la mantisa toma un valor negativo. Si lo negativo es la mantisa, el efecto es diferente, pero veámoslo mediante la utilización del programa.

ALMACENAMIENTO DE NUMEROS

El conocimiento de los números exponenciales puede ser interesante, pero también son útiles para mejor conocer la forma en que la mayoría de los ordenadores almacenan los números.

Cuando tecleas un comando tan simple como PRINT 10*10, el ordenador los trata realmente con más detalle. Traduce los números en un tipo de formato exponencial llamado «coma flotante» (punto flotante para los anglosajones) y los calcula con esta forma.

Existe también otra complicación. Como podrás ver en el artículo destinado a la creación de GDUs (Gráficos Definidos por el Usuario) desde el BASIC, los ordenadores almacenan cada número en binario, o base dos.

Para mostrar cómo se guardan los números en la RAM de tu ordenador, sigue este ejemplo de un cálculo basado en el número 10.

El primer paso es convertirlo a binario, lo que da los números 00001010,00000...

Los cuatro primeros ceros de este número en binario no significan nada, y podría fácilmente haberse prescindido de ellos, lo que nos deja con 1010,00000... Como vimos previamente, un número decimal con exponente consiste en un número situado entre 1,0 y 9,999999... Cuando se almacena un número en coma (punto) flotante en el ordenador, la mantisa consiste en un número binario que siempre comienza en .1.

Esto es posible debido a que el ta-

maño de la parte exponencial del número determina la posición de una «coma binaria», el equivalente binario de la coma decimal. Como es algo definido automáticamente, la mantisa no necesita especificar donde se situa la coma.

Existe un problema: ¿cómo ajustas el exponente para mostrar donde debería estar el punto binario, y como conviertes el número binario para que comience en .1?

DESPLAZANDO LA COMA

En efecto, la respuesta a ambos problemas es la misma. Todo lo que harás es desplazar la coma hasta que se enfrente al primer 1 y por cada lugar que se mueva la coma añadirás un 1 al exponente. Por ejemplo, con el número decimal 58 (111010.000 en binario), la coma binaria se mueve gradualmente hacia la izquierda, hasta que no haya números a su izquierda, únicamente ceros. En este caso has desplazado la coma 6 lugares a la izquierda, por lo que el exponente es +6 y la mantisa .11101000...

En efecto, el exponente comienza valiendo 128, por razones que descubrirás más tarde, le sumarás el anterior número. De esta forma, el exponente del ejemplo con el número 58 se convierte en 128+6= 134, que el ordenador almacena en forma binaria.

Para recapitular, el ordenador almacena la parte exponencial del número en un byte y también la mantisa, que puede utilizar hasta 4 bytes, tres de los cuales serían llenados con ceros en este ejemplo.

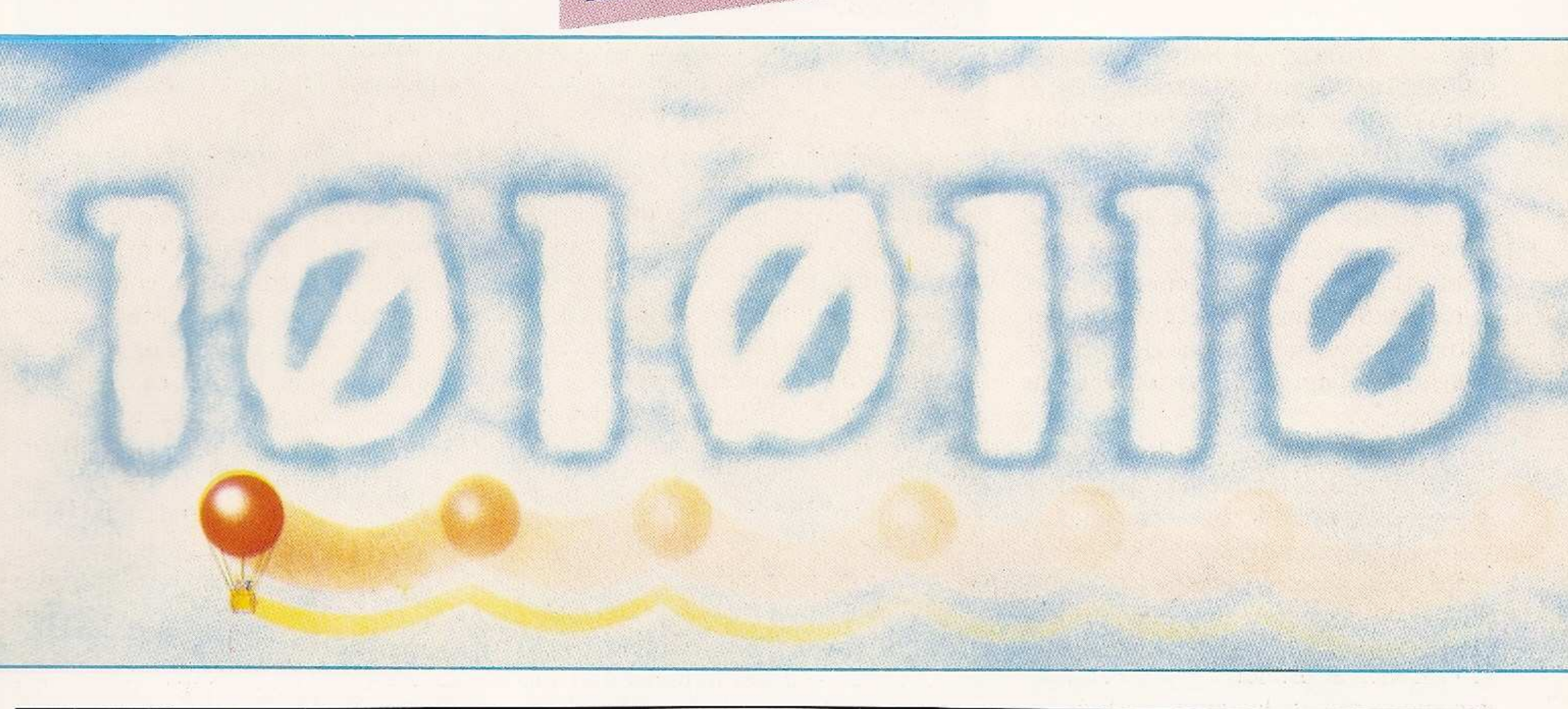
Cualquier número en coma flotante puede ocupar hasta cinco bytes . Esto limita el tamaño del número máximo que puede almacenarse en el ordenador. El único byte dedicado al exponente da un máximo posible de 2 elevado a la potencia 127. Podría ser hasta 255, pero el primer bit es utilizado para el signo del exponente.

El máximo número utilizable por la mantisa es .11111 etc, con todos los bits valiendo 1. esto se aproxima mucho a 1.000 etc., que es el valor 1 tanto en binario como en decimal. El menor número es .1000 etc. siendo todos 0 excepto el primer bit. El primero ha de ser un 1 puesto que la coma (punto) binaria se desplaza hasta alcanzar el primer 1 del número. Por otra parte, conviene saber que .1 es el equivalente binario de 1/2 en decimal.

Multiplicando las dos partes del número, puedes hallar el mayor número almacenable en formato de coma flotante, que es 1.70141E38. Puedes emplear el programa de más adelante para comprobar cómo lo almacena el ordenador.

CASOS ESPECIALES

Ahora que tienes una completa idea del proceso, veremos que existen dos variaciones a ello.



La primera viene cuando el número inicial es menor que uno. En este caso el desplazamiento de la coma hacia la izquierda hace que vaya en sentido contrario de como debiera ir. Por lo que habrás de dirigirla hacia la derecha. Así el ordenador sabe donde debería situarse la coma en el número final, sustrayendo uno del exponente cada vez que se retrocede. El exponente comienza, como con los números mayores que 1, con un valor de 128.

A menudo, cuando escribes un número binario, se escriben sobre ellos los equivalentes decimales de cada «columna». Volvemos a remitirnos a la conversión binario decimal del artículo de los GDUs. Luego se suman los valores de las columnas que tienen un 1 y se obtiene el equivalente decimal. Son los valores 1,2,4,8,6,32,64..., pero recordemos que estos se situan a la izquierda de la coma. En el caso de que la coma binaria se encuentra a la derecha, los valores asociados a cada columna son fracciones: 1/2, 1/4, 1/8, 1/16, 1/32, 1/64...

La segunda variante con los números en coma flotante se produce cuando el número debe ser capaz de almacenar su propia versión del signo menos de alguna forma, sin utilizar espacio adicional alguno de la memoria.

Se vuelve a recurrir al truco de emplear el primer bit para saber si el número es positivo o negativo. Simplemente, si es negativo el primer bit de la mantisa será puesto a 0.

¿DONDE ESTA EL NUMERO?

Con los ejemplos anteriores en mente, ahora estás en disposición de ser capaz de saber como almacena los números tu ordenador.

El número decimal 58 (111010 en binario) es almacenado con los siguientes 5 bytes:

134 104 0 0 0

y el número 10 como:

132 32 0 0 0

Haz algunos experimentos, calculando los valores decimales de los bytes que el ordenador almacena para tus números. Comprueba tus respuestas copiando y ejecutando el programa que sigue:

Teclea para C-64 y Vic-20

```
10 PRINT"[SHIFT+CLR/HOME]":
   CLR
20 PRINT "ESCRIBE UN NUMERO"
   :INPUT X
30 V=PEEK(45)+PEEK(46)*256
40 PRINT "EXPONENTE:";PEEK
   (V+2)
50 PRINT "MANTISA:";:FOR N=3
   TO 6
60 PRINT TAB(10);PEEK(V+N)
70 NEXT N
80 END
```

El programa siguiente complementa al anterior:

```
100 REM***SEGUNDO PROGRAMA
110 PRINT"s":CLR:R=1:V=PEEK
    (45)+PEEK(46)*256
120 INPUT"ESCRIBE EL EXPO
    NENTE";EX
130 IF EX<0 OR EX>255 THEN
    120
140 POKE V+2,EX
150 FOR N=3 TO 6
160 INPUT "ESCRIBE LA
    MANTISA ";MAN
170 IF MAN<0 OR MAN>255 THEN
    160
180 POKE V+N,MAN
190 NEXT
200 PRINT "RESULTADO=";R
```

El primero de los dos programas te permite introducir un número, devolviendo el ordenador los cinco bytes utilizados para almacenarlo.

El segundo trabaja en sentido inverso, introduciendo los 5 bytes se obtiene el número que representa. Algunas veces la respuesta presentará el formato exponencial.

AHORRANDO MEMORIA

Conociendo la necesidad de los 5 bytes para cada número, pronto te das cuenta de que a menudo se des-

perdicia memoria al haber números que no necesitan totalmente los 4 bytes de la mantisa. Podrían ser ahorrados y tal vez utilizados para algunas otra cosa.

En efecto, puedes ahorrar una cantidad significativa de memoria evitando el uso de cantidad de números, pero esto no es siempre posible. Sin embargo hay otra forma de reducir la cantidad de memoria para cada uno.

Se puede evitar en muchos casos el uso de números, simplemente estableciendo una variable que sea igual al valor que deseas utilizar. Naturalmente que precisas del número en sí para definir la variable y ésta también necesita algunos bytes de almacenamiento. Si vas a utilizar ese número una o dos veces, probablemente no merezca la pena.

Una vez que se establece una variable, necesitarás solamente un byte para cada caracter empleado por ella, por lo que si recurres a variables de una sola letra, es probable que obtengas un gran ahorro.



EFECTOS EXTRAÑOS

Una vez comprendido todo lo anterior, podrás deducir porque se producen a veces cosas raras. Existen imprecisiones de cálculo que algunas personas tienden a asociar con errores en la ROM del ordenador. La razón de que aparezcan o desaparezcan cifras aparentemente de un modo aleatorio, se debe a que el ordenador trabaja con algún grado de imprecisión, y el ordenador tiende a redondear con números que poseen más de un determinado número de dígitos. Esto depende de cuales sean los números y cuantos cálculos subsecuentes se realizan con los valores redondeados. Esto puede conducir a resultados erróneos en algunos casos.

Afortunadamente, los ordenadores son suficientemente precisos para la mayoría de las aplicaciones que se los ocurren.

Utilizando matrices se pueden almacenar números con más cifras significativas que el máximo de nueve que permite el ordenador. La principal ventaja consiste en que puedes escribir una rutina que visualice el número en su forma normal y no con exponente. Con este método puedes también solventar el problema de mensajes de error tales como *«number too large»*.

:RITEMAN: news

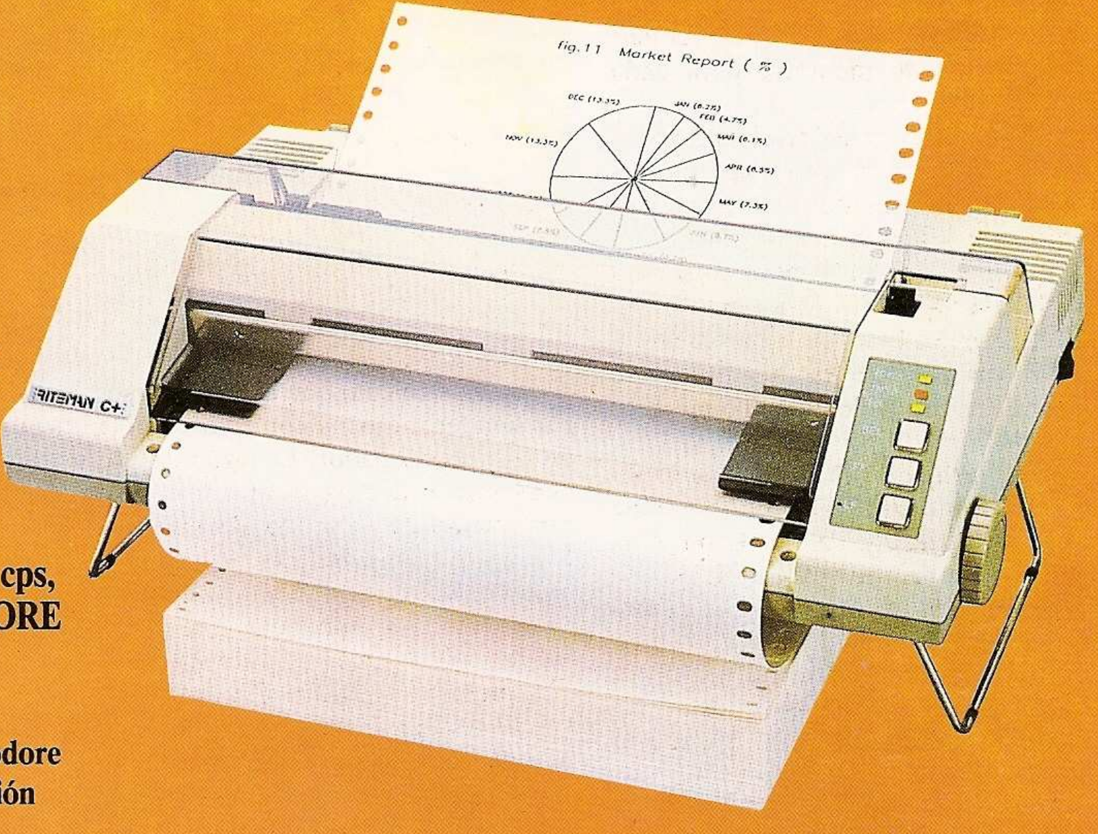
DATAMON REPRESENTACION EN ESPAÑA DE: :RITEMAN:

PROVENZA, 385-387
TEL. (93) 207 24 99*

TELEX 97791
08025 BARCELONA

IMPRESORA PARA SU COMMODORE
(óptima relación precio/prestaciones)

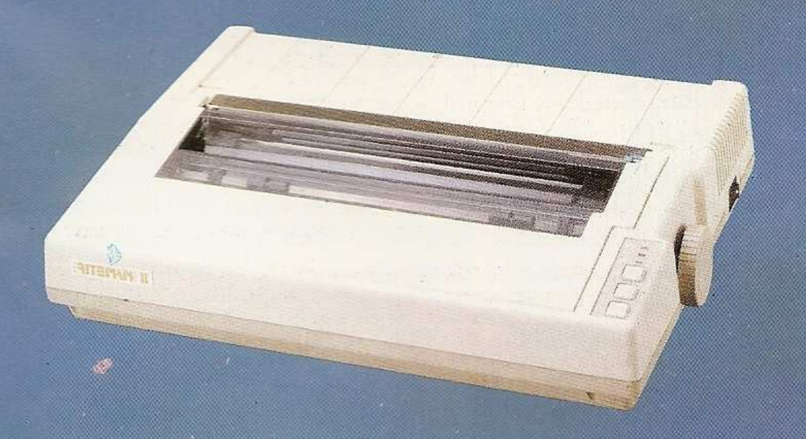
—Cabezal 9 agujas
—Doble operatividad
—Cinta autoretintada
—Tampón retintable
—Ausencia de rodillo
—No dobla el papel
—Elevadores inferiores
—Admite texto rígido
—Máximos tipos de escritura



Modelo SUPER C+, 120 cps, NLQ, ASCII y COMMODORE

• **Conexión directa a Commodore (cable incl.) Tracción y fricción**

LA IMPRESORA PARA COMMODORE, ASCII Y PC'S COMPATIBLES
(Máxima versatilidad/precio ajustado)



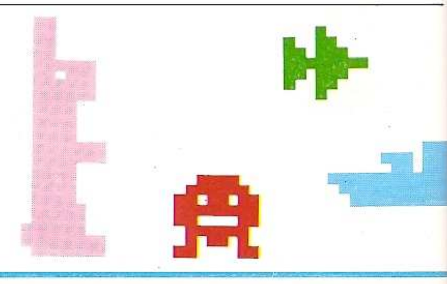
RITEMAN 10-C

—140 cps, tracción y fricción
—Paralelo centronics/Commodore serie DIN
—Tablas ASCII y PC en Rom interna
—Tabla 100% Commodore y 8K RAM en módulo
—Interface Commodore exterior incluido
—RS 232-C opcional

NOTA: Para Aplicaciones en las que se necesite más velocidad, o mayor tamaño de carro, también pueden aplicarse nuestros interfaces externos a los modelos RITEMAN 10/II y RITEMAN 15.

• **Estaremos en S.I.M.O. Pabellón IX Stand. G. 104.**

CARACTERES GRAFICOS INMEDIATOS



No es preciso conocer el código máquina o ser un experto en el manejo del sistema binario para desarrollar gráficos utilizables en juegos. Con estos sencillos ejercicios de programación es posible realizarlos inmediatamente.

En tan sólo diez minutos es posible crear caracteres gráficos originales para emplearlos en el diseño de juegos propios. Todos los ordenadores domésticos poseen su propio método para producir caracteres gráficos nuevos o GDUs (Gráficos Definidos por el Usuario). En el **Commodore 64** existen unos muy especiales, los conocidos *sprites* con dimensión de 24 × 21 *pixels*, que no son el objetivo de este artículo.

Sin embargo, en muchos casos es conveniente partir de símbolos cuya dimensión es de 8 por 8 *pixels*, generalmente para los enemigos en un juego de marcianos. Una vez adquirida cierta experiencia es sencillo crear imágenes gráficas de mayores dimensiones, uniendo varios símbolos para crearlas.

EL DISEÑO EN BINARIO

Como es sabido, los ordenadores memorizan toda la información en formato binario (o sea en base 2). Aunque no es preciso conocer el manejo del sistema binario para transformar los caracteres formados sobre papel cuadriculado en una serie de unos y ceros. Todo lo que conviene saber es:

1.- Cada vez que se quiere iluminar un punto (*pixel*) se le asocia un 1.

2.- Cuando el punto no debe aparecer se emplea un 0. Si se dibuja, por ejemplo, la cruz de Lorena mostrada más adelante, la primera línea consiste en tres espacios, un punto y otros cuatro espacios. Esto en binario es 00010000. La segunda línea se compone de dos espacios, tres puntos y tres espacios: 00111000. La imagen completa se representa así:

0	0	0	1	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	0	0	0	0
0	1	1	1	1	1	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

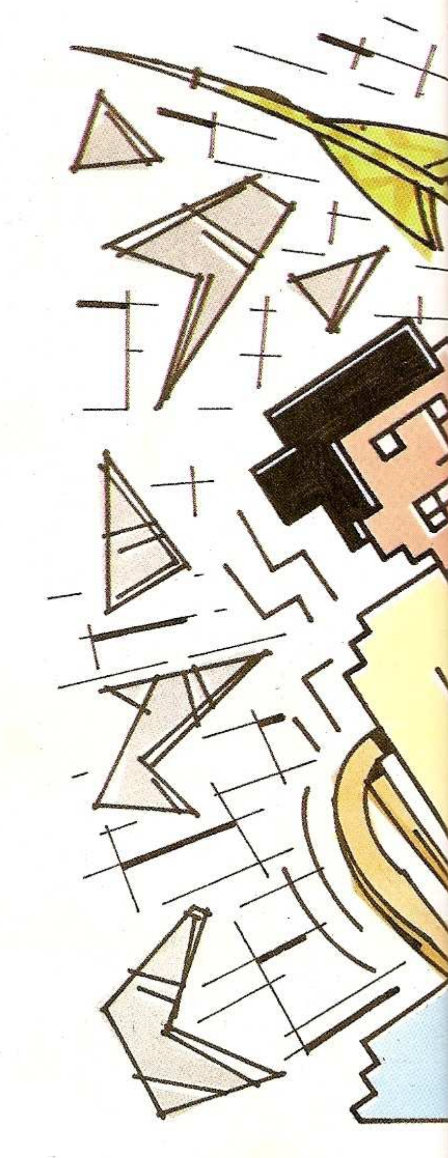
El programa de este artículo se encarga de todas las operaciones a partir de la información de unos y ceros.

PASO DE BINARIO A DECIMAL

El camino más corto para convertir los numéros binarios en decimales consiste en recurrir a una tabla de 8 columnas y nueve filas. Cada columna lleva asociado un valor, potencia de 2 (elevado a su posición): 128, 64, 32, 16, 8, 4, 2, 1. El resto de las líneas se llenan con las series de valores binarios asociados a la imagen a formar. Siguiendo con el ejemplo de la cruz de Lorena:

128	64	32	16	8	4	2	1
0	0	0	1	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	0	0	0	0
0	1	1	1	1	1	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

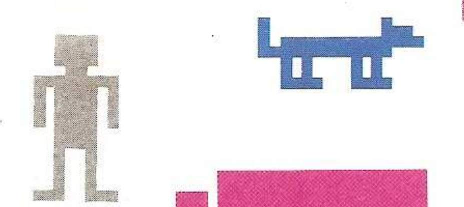
Para realizar la conversión, se ignoran todos los ceros. En cada fila horizontal se suman los valores asociados a cada columna en la que exista un uno presente. En el ejemplo, la primera fila arroja un valor de 16, la segunda es de 56 (32+16+8).



Realizando las operaciones en todas las filas obtenemos una serie de 8 valores, que podrían añadirse en una sentencia DATA de la siguiente apariencia:

```
DATA 16,56,16,124,16,16,16,0
```

Inicialmente puede parecer un método algo incómodo, pero resulta, sin embargo, bastante rápido. Por otro lado hay muchas combinaciones recurrentes, tales como 11111111 (255 en decimal), que resultan fáciles de recordar.



- DEL DISEÑO AL SISTEMA BINARIO
- PASO DE BINARIO A DECIMAL
- BINARIO A HEXADECIMAL
- PROGRAMA PARA C-64

BINARIO A HEXADECIMAL

La conversión de binario a hexadecimal es bastante fácil. Con la ayuda de la siguiente tabla se realiza sin mayores complicaciones:

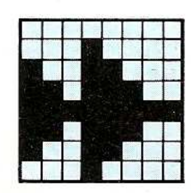
```
Binario              Hexadecimal
-------              -----------
0000.......................0
0001.......................1
0010.......................2
0011.......................3
0100.......................4
0101.......................5
0110.......................6
0111.......................7
1000.......................8
1001.......................9
1010.......................A
1011.......................B
1100.......................C
1101.......................D
1110.......................E
1111.......................F
```

En este caso no pueden ser ignorados los ceros. Lo primero que debemos hacer es agrupar los 8 bytes en 2 conjuntos de 4 bits (*nybble*). A continuación se busca en la tabla el equivalente hexadecimal del número binario de cuatro cifras. Se anota y vuelve a repetirse la operación. El número obtenido por este procedimiento será el correspondiente hexadecimal del binario propuesto. Si volvemos al ejemplo de la cruz de Lorena, la primera fila es: 0001 0000 = $10 (el signo $ suele utilizarse para denotar el caracter hexadecimal). La segunda línea es: 0011 1000 = $38, etc.

Repitiendo el proceso nos encontra-

mos con una serie de números: 10,38,10,7C,10,10,10,00.

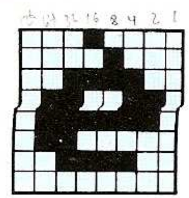
Naturalmente que es posible escribir un programa que haga estas operaciones automáticamente, pero ese no es el fin del artículo.



APLICACION CONCRETA

Realmente en el **Commodore 64** se utilizan los caracteres gráficos menos frecuentemente que los versátiles *sprites*, pero conviene saber definir GDUs de 8 por 8 *pixels* como un recurso de gran utilidad. En particular es posible redefinir total o parcialmente el juego de caracteres originalmente utilizados por el **64**. Por poner un ejemplo próximo, es factible definir las letras minúsculas acentuadas, que no existen en el repertorio del ordenador. En el ejemplo diseñaremos una e acentuada, que siguiendo el dibujo sobre papel cuadriculado, arroja los siguientes valores:

Binario	Hex.	Decimal
00010000	10	16
00001000	08	8
00111100	3C	60
01100110	66	102
01111110	7E	126
01100000	60	96
00111100	3C	60
00000000	00	0

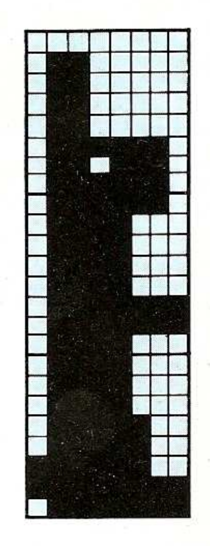


El esquema decimal es el que ofrece un uso más inmediato y fácil de calcular. Sin embargo no descartamos que puedan emplearse los binarios y hexadecimal.

Procedamos a incluir esta letra en nuevo juego de caracteres.

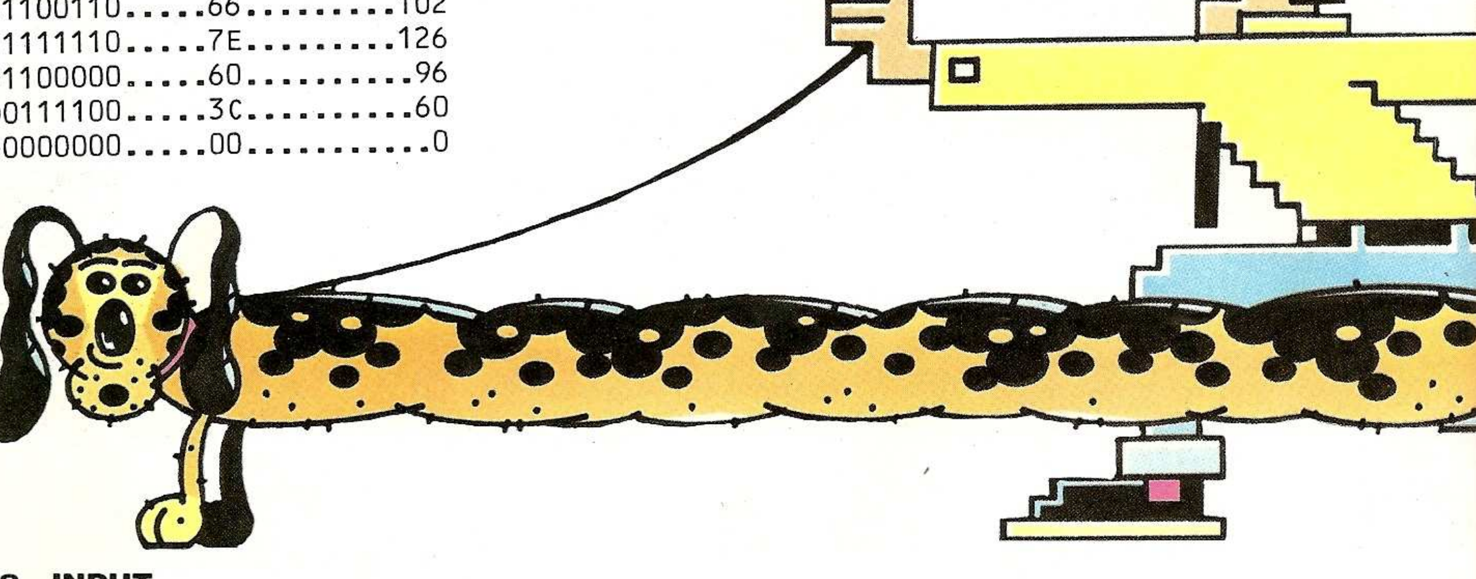
```
Teclea para C-64

10 A=12:Z=A*1024/256
20 POKE 53272,(PEEK(53272)
```

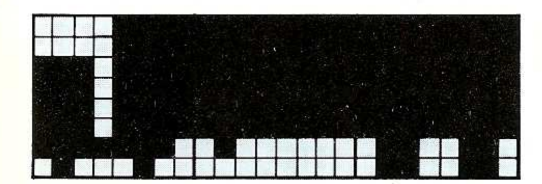


```
   AND 240)OR A
30 POKE 52,Z:POKE 56,Z:CLR
   :A=12
40 POKE 56334,PEEK(56334)
   AND 254
50 POKE 1,PEEK(1) AND 251
60 FOR J=0 TO 56832-53248
70 POKE A*1024+J,PEEK(53248+
   J)
80 NEXT J
90 POKE 1,PEEK(1)OR 4
100 POKE 56334,PEEK(56334)
    OR 1
110 SC=5:Z=1024*12:FOR J=Z+
    (SC*8) TO Z+(SC*8)+7:
    READ A$
120 N=0:FOR T=1 TO LEN(A$)
130 IF MID$(A$,T,1)="1" THEN
    N=N+2^(LEN(A$)-T)
140 NEXT:POKE J,N:NEXT J
500 DATA 00010000
510 DATA 00001000
520 DATA 00111100
530 DATA 01100110
540 DATA 01111110
550 DATA 01100000
560 DATA 00111100
570 DATA 00000000
```

Una vez que se ejecuta el programa, no sucederá nada durante aproximadamente un minuto, pero cuando aparece el mensaje READY las letras E aparecerán sustituidas por el nuevo

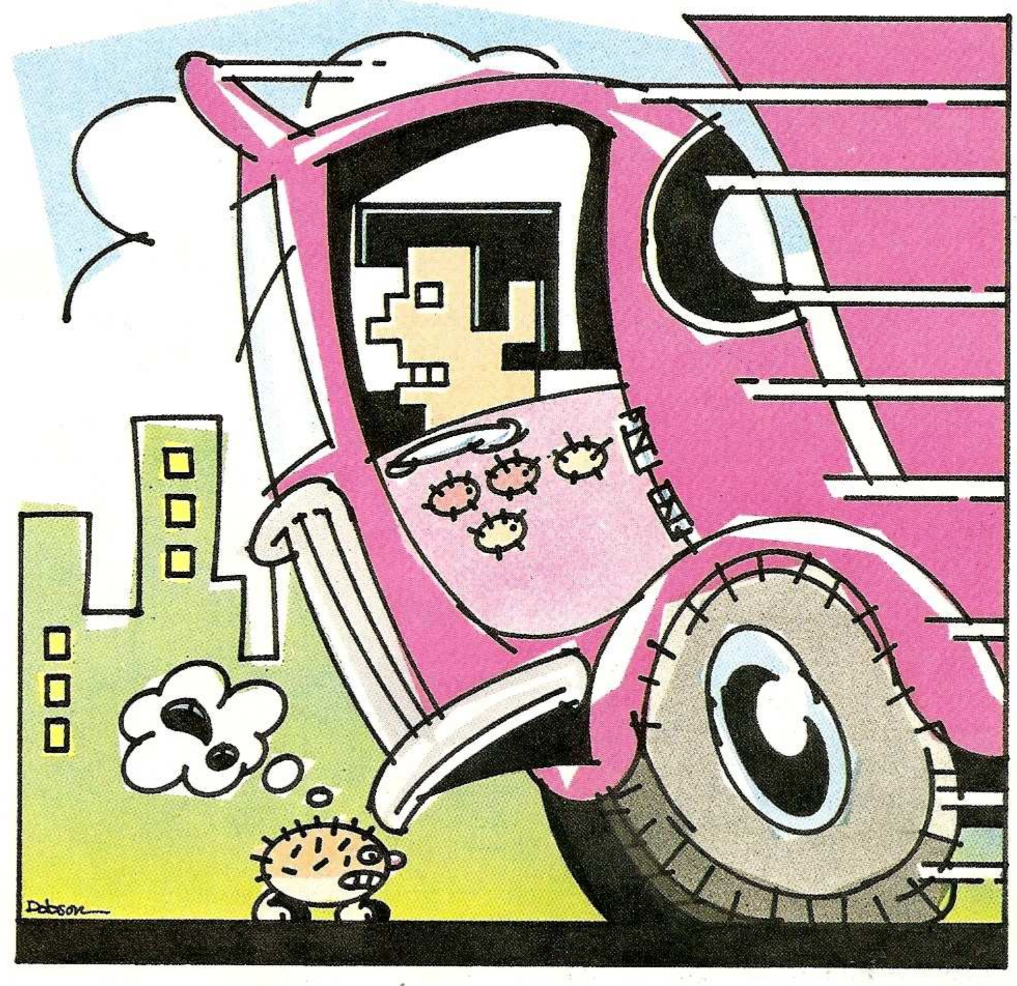


caracter. Lo mismo ocurre al presionar la tecla correspondiente a la letra. Si presionamos las teclas Commodore y SHIFT las E perderán su acento al pasar al segundo juego de caracteres del **64**.



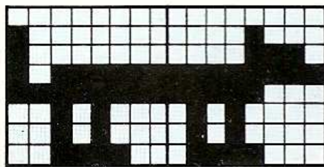
Para crear cualquier otro símbolo gráfico asociado a la E, no hay más que alterar las series de unos y ceros en las líneas con DATA.

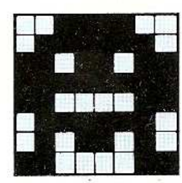
Si cambiamos el valor de la variable SC en la línea 110, será otro el caracter redefinido. Por ejemplo, un valor de 6 cambiará la F, el 7 la G, etc. En tal caso, convendrá escribir, GOTO 110, en lugar de RUN, que repetiría el lento proceso de copia del juego de caracteres desde la ROM a la RAM que ocupa las mismas direcciones de memoria.



DESCRIPCION DEL PROGRAMA

La información referente a los caracteres normales está almacenada en memoria ROM y por tanto no puede ser alterada. Sin embargo, la técnica descrita se basa en copiar esta información en memoria RAM, sea total o parcialmente. De esta manera ya es fácil manipular los bytes que definen el aspecto de las letras, símbolos y demás caracteres gráficos.





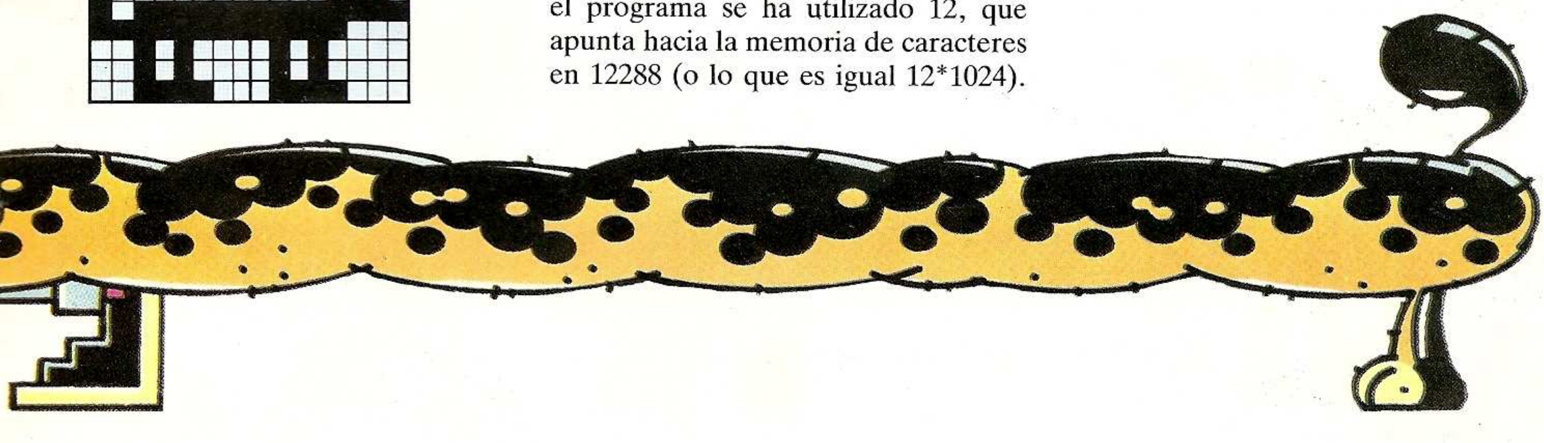
La primera operación que acomete el programa consiste en asignar una cantidad de RAM suficiente para contener al nuevo conjunto de caracteres. El valor de A en las líneas 10 y 30, permite definir el área de memoria a utilizar. Se puede utilizar cualquier valor entero comprendido entre 4 y 16. En el programa se ha utilizado 12, que apunta hacia la memoria de caracteres en 12288 (o lo que es igual 12*1024).

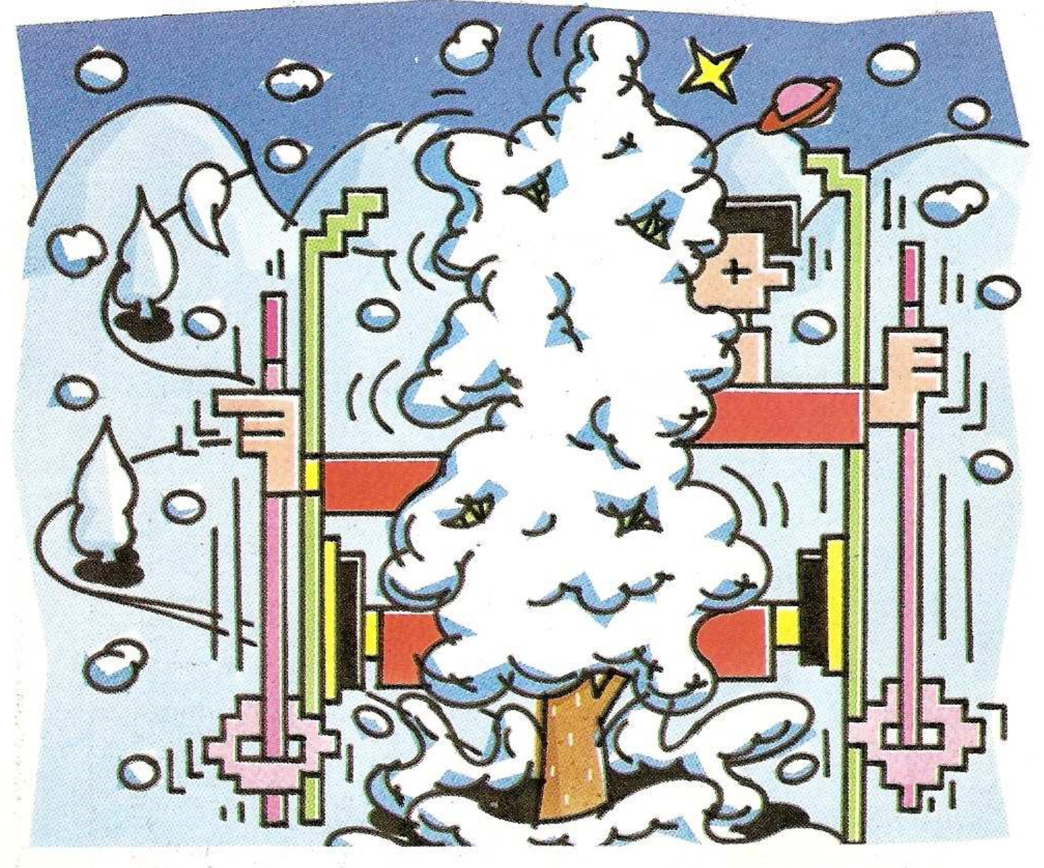
Un valor de 4 situaría el puntero en la dirección 4096 (4*1024).

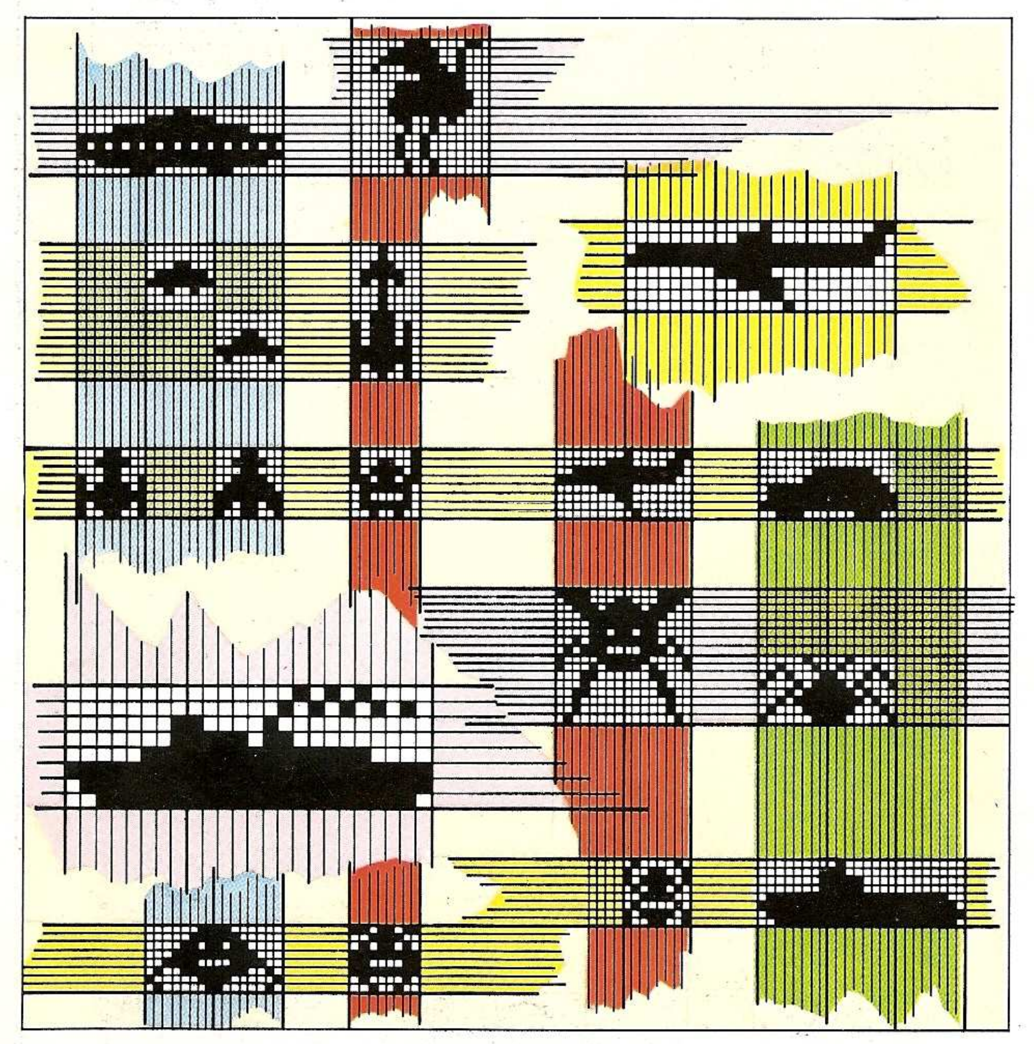
La línea 30 ajusta dos punteros, el final del área destinada al BASIC y el inicio del área de cadenas. Con esto se consigue que el nuevo conjunto de caracteres no se superponga a los programas en BASIC deteriorándolos.

En la línea 40 se desactiva la llamada «interrupción para exploración del teclado». La línea 50 selecciona la ROM de caracteres.

La rutina de copia desde la ROM a



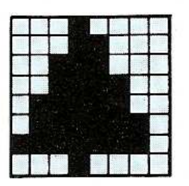




la RAM interviene en las líneas 60,70 y 80. Los números 53248 y 56832 se refieren al comienzo y final de los grupos de ocho bloques de caracteres que se copian en la RAM. El programa realiza así una copia integral, cambiando el arco de valores atribuido a J.

Una vez completa la copia, se desactiva el generador de caracteres de la ROM (línea 90), al tiempo que se reactiva la exploración del teclado (línea 100).

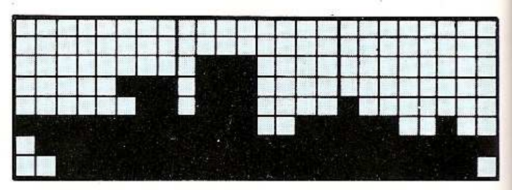
Las líneas 110 a 140 cambian la definición de un caracter dado (el SC), leyendo la serie de valores contenidos en las sentencias DATA (líneas 500 a 580), convirtiéndolos en valores decimales (líneas 120 y 130).



Para ahorrar espacio, las líneas DATA pueden ser sustituidas por una sóla:

```
500 DATA 00010000,00001000,
    00111100,01100110,01111
    110,01100000,00111100,
    00000000
```

Es notable el ahorro, pero este formato no permite evaluar fácilmente el aspecto que tendrá el caracter ni corregirlo sobre la marcha.



Si se prefiere trabajar en hexadecimal, habrá que realizar la siguiente modificación:

```
130 M=ASC(MID$(A$,T,1))-48:
    N=(M+(M>9)*7)*16^(LEN
    (A$)-T)+N
500 DATA 10,08,3C,66,7E,60,
    3C,00
```

ESTRENO MUNDIAL

THE FINAL CARTRIDGE 2®

EL PRIMER SISTEMA OPERATIVO EXTERNO PARA EL CBM 64

AHORA, CON FREEZER COMPLETO

(Puede cargar sus backups sin el cartucho)

Este sistema operativo construido en un cartucho no utiliza memoria alguna y está siempre presente. Compatible con 99% de los programas.

INCORPORA

TURBO DISCO: Carga 6 veces más rápido -salva 6 veces más rápido - No Borra la pantalla.

TURBO CINTA: 10 veces más rápido, incluso con ficheros. Utiliza los comandos normales del Commodore (Load-Save-Inputπ,...)

INTERFACE CENTRONICS: Permite utilizar las impresoras de tipo paralelo. Imprime los gráficos Commodore y los códigos de control.
(Importante para los listados).

VOLCADOS DE PANTALLA: De alta revolución y texto. 1 página de ancho. 12 tonos grises, copia pantallas de juegos o de programas como Doodle, Koala Pad, Print Shop, etc. Busca automáticamente la dirección en memoria.
Funciona con impresoras Commodore y Centronics.

24 K. MAS DESDE EL BASIC: 2 nuevos comandos, "memory write" y "memory read" mueven 192 bytes muy rápidamente en cualquier sitio de los 64 RAM del CBM 64. Se pueden usar con cadenas variables.

COMANDOS DEL BASIC 4.0: Como dload, dsave, dappend, catalog, etc.

TECLAS DE FUNCION TELEPROGRAMADAS: Run, load, save, catalog, comandos de disco, list (quita las protecciones contra el listado de los programas en BASIC).

MONITOR DE CODIGO MAQUINA: Scroll hacia arriba y abajo. Bankswitching (para leer y escribir debajo de las ROOMS), etc... No reside en memoria. Se puede llamar en cualquier momento con cualquier programa en memoria.
RESET: Resetea todos los programas.

TRAINING MODE: Para cancelar la detección de colisión de sprites podrás ir hasta el final de los juegos sin que te maten los bichos malos.





Ejemplo de volcado de pantalla.

¡Hasta el precio es increíble!



ATENCION: Las copias conseguidas con THE FINAL CARTRIGE 2 son exclusivamente para uso propio.

SUPER FREEZER

Pulsando el botón del FREEZER, tomará el control de su ordenador, "congelando" el programa en memoria. Con la ayuda de un menú muy cómodo podrá:

—Hacer VOLCADOS DE PANTALLA, alta o baja resolución (por ejemplo las pantallas de sus juegos preferidos). Podrá "congelar" cualquier programa en el momento que desee, y volcar la pantalla sobre el papel.

—Hacer COPIAS DE SEGURIDAD de sus programas. El FREEZER le permitirá, con programas protegidos o no, cualquiera que sea el sistema de carga utilizado (turbo, verificación de errores, entre pistas...).

- Hacer copias de cinta a cinta
- Hacer copias de cinta a disco
- Hacer copias de disco a cinta
- Hacer copias de disco a disco

sólo pulsando una tecla. El proceso de copia es totalmente automático, y no necesita tener ningún conocimiento de programación. La copia facilitada por el FREEZER consta sólo de dos partes (un cargador y el programa propiamente dicho) y se puede cargar. INDEPENDIENTEMENTE DEL CARTUCHO (a velocidad turbo), EL FREEZER de THE FINAL CARTRIDGE 2 es más cómodo y más rápido que los productos especializados ingleses o americanos probados hasta ahora.

NO EXISTE NINGUN PRODUCTO COMPARABLE PARA SU C64: Encontrará quizás algún FREEZER (con otro nombre) inglés o americano, pero además de ser más caro será sólo un FREEZER.
THE FINAL CARTRIDGE 2 da mucho más, por menos.

DISPONIBLE EN LAS MEJORES TIENDAS O DIRECTAMENTE POR CORREO O TELEFONO

Condiciones especiales para distribuidores



EL PROCESAMIENTO DE TEXTOS

- NOMBRES DE FICHEROS
- EDICION
- AJUSTE DE COLORES
- PARAMETROS DE ENTRADA Y SALIDA
- MANIPULACION DEL TEXTO

Si quieres poner a punto tu habilidad mecanográfica a expensas de un procesador de textos hecho y derecho, la respuesta la tienes en este editor de textos de fácil programación.

Ya conoces las ventajas del procesado de textos. Realmente ésta es una de las aplicaciones más útiles de los ordenadores domésticos. Para conseguir los mejores resultados, puedes comprarte el *software* existente en código máquina, pero normalmente está diseñado para usos profesionales y puede resultar muy caro.

La parte de programa que presentamos en este artículo presenta un buen editor de textos para quien da sus primeros pasos en este terreno, permitiéndole adquirir una sensación real de cómo son los programas de este tipo.

Con este programa puedes crear informes, cartas o cualquier otra forma de correspondencia. Naturalmente la salida se hace a través de la impresora. El texto puede ser almacenado en cinta o disco y puede recuperarse para su posterior utilización, ya sea por este programa o cualquier otro que haga uso de los ficheros secuenciales creados con los datos.

Debido a la longitud del programa completo —no menos de 7,5K— el programa se ha fraccionado en dos unidades para hacer más fácil su digestión, y no funcionará adecuadamente hasta que se haya introducido la última parte. Sin embargo, después de introducir la primera parte ya estarás en condiciones de crear ficheros de texto. La segunda parte contiene la rutina de impresora, que evidentemente necesitarás para obtener copias de tus escritos. También se incluyen unas facilidades de ordenación y búsqueda, así como una opción para hacer modelos de cartas.

La extensión del programa hace que no pueda ejecutarse con el **Vic 20** sin ampliación de memoria. Debido a esto, y a que la pantalla del **Vic** sólo presentaría una pequeña cantidad de texto, no incluiremos la versión para el **Vic**.

Antes de introducir la primera parte del programa, echemos una ojeada a la forma en que se utiliza. Se ha configurado de una forma que resulte de uso cómodo en la mayoría de las aplicaciones domésticas, pero como ocurre con todo programa de este tipo puedes modificar el contenido de los diversos menús y mensajes de pantalla para que se adapten a tus necesidades o a tus preferencias. No obstante, si decides realizar dicha modificación, deberás respetar las mismas convenciones de numeración de líneas para que el programa se enlace correctamente con otros módulos, especialmente cuando se le añadan otros módulos.

EL MENU PRINCIPAL

En cuanto ejecutes (con RUN) la versión completa de la copia del programa, te aparecerá un mensaje en pantalla invitándote a seleccionar las opciones de entrada y salida. Con ellas el sistema queda configurado para utilizar cinta o disco, lo cual se hace tecleando C ó D.

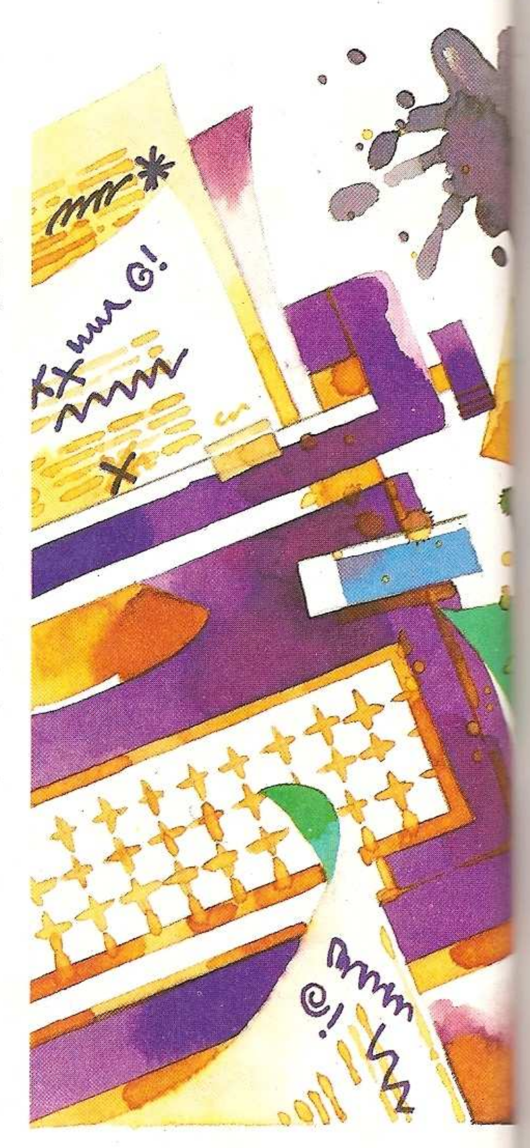
Observa cómo puedes utilizar un dispositivo de entrada y otro distinto de salida, y cambiar después dichos dispositivos cuando quieras.

Seguidamente te aparecerá el menú general.

[S]AVE (Guardar)
[C]LEAR (Borrar)

LOAD

Si quieres seguir trabajando sobre un fichero de texto ya existente, que anteriormente habías guardado (con el comando SAVE), pulsa L para arrancar la rutina de carga. Como el texto se va a escribir sobre cualquier texto que tengas en memoria, el programa te enviará un mensaje de «Estás seguro?». Para continuar, pulsa Y; si quieres abortar el proceso, pulsa cualquier otra tecla. Seguidamente te pedirá el nombre de un fichero y tendrás que proporcionárselo para que el proceso continúe. El fichero de texto cuyo nombre hayas tecleado, será cargado en la memoria y ya puedes trabajar sobre él a tu antojo.



SAVE

La rutina SAVE se utiliza para la creación de ficheros secuenciales para almacenamiento de datos. Cuando selecciones esta opción pulsando la tecla S, lo primero que te pedirá el programa es un nombre de fichero. No se aceptará una entrada nula. El sistema ya «sabe» el dispositivo que estás utilizando y continúa enviándote mensajes en consecuencia, acerca de la preparación de la cinta o la unidad de disco. Por último, pulsa RETURN para que se efectúe la operacion de almacenamiento.

Para almacenar los ficheros en disco, tendrás que utilizar el procedimiento de escribir «@0:nombre» excepto en los casos en que no importe que el fichero de texto se escriba sobre el último con el que trabajaste.

Se pueden añadir otras rutinas adicionales de SAVE que permitan la protección de los ficheros, el cambio de nombre o su sustitución. En el listado que presentaremos en nuestro próximo artículo podrás examinar varias rutinas de muestra.

ENTRADA/SALIDA

Si por cualquier razón decides modificar los parámetros iniciales de entrada/salida, puedes hacerlo en cualquier momento seleccionando la opción I en el menú principal. Teclea tu elección y automáticamente se seleccionará el programa para la nueva configuración.

MODO DE EDICION

Si seleccionas E en el menú principal, entras en el modo de edición y te aparecerá un menú secundario.



Puedes seleccionar una de las primeras opciones para trasladarte al principio o al final del texto que tienes en memoria. Si utilizas la tercera opción, puedes volver a la última línea editada o salir al menú principal.

Con la opción COLOR puedes ajustar a tu gusto la «tinta», el «papel» y el borde de pantalla.

Cada una de las tres primeras opciones te sitúa en un modo de entrada. La pantalla presenta las líneas del principio o el final del texto (sólo puede presentar a un tiempo una de las dos cosas si ya tienes algo en memoria). Cerca de la parte inferior de la pantalla hay una «zona de trabajo» separada.

El texto introducido en la zona inferior se transfiere a la memoria y a la zona de pantalla cuando se pulsa RETURN, o de forma automática cuando se han completado dos líneas de pantalla.

El programa lleva incorporadas varias utilidades de edición. Toda la labor de edición e introducción tiene lugar en la zona de trabajo.

Los controles de edición te permiten desplazarte hacia adelante y hacia atrás a lo largo de la línea de texto del área de trabajo para insertar o suprimir caracteres antes de que el texto sea transferido a la memoria.

En otras palabras, el texto que ya está en memoria, representado en la parte alta de la pantalla, tiene que ser copiado línea a línea a la zona de trabajo para poder trabajar sobre él.

Puedes introducir nuevas líneas en cualquier parte del cuerpo principal del texto. Para hacerlo, pulsa la tecla adecuada del modo «editor» (consulta las instrucciones detalladas) y coloca la marca debajo del punto donde quieras insertar la nueva línea.

También puedes suprimir texto línea a línea entrando en el modo editor y colocando nuevamente la marca debajo de la línea que quieras borrar antes de pulsar la tecla adecuada de borrado.

Las teclas de control te permiten saltar hacia adelante o hacia atrás en el texto que tienes en memoria, de diez en diez líneas.

Para volver al menú del modo editor, pulsa la tecla de escape adecuada y desde allí al menú principal para modificar cualquiera de los parámetros del sistema.

BORRADO

Otra de las funciones disponibles en el menú principal es el borrado de la memoria, pero sólo después de que hayas respondido «si» al mensaje «Estás seguro?» que te aparecerá en pantalla en cuanto pulses la tecla B. Ten en cuenta que esta opción no inicializa los parámetros de entrada/salida ni la impresora. Si quieres arrancar de nuevo desde cero, pulsa F para salir totalmente del programa, con lo que se produce una inicialización completa del sistema al ordenador al BASIC.

Cuando hayas tecleado la parte de programa que figura al final del artículo, podrás hacer pleno uso de las facilidades de edición que dicho programa te ofrece. Pero recuerda que no podrás imprimir ningún texto hasta que hayas introducido la última parte del programa, que publicaremos en el siguiente capítulo de esta breve serie.

Como el arte de escribir cartas se ha convertido en una cosa del pasado, encontrarás que las facilidades de edición que ofrece el editor de textos resultan de gran ayuda a muchos niveles. En el nivel más sencillo, puedes hacer correcciones ortográficas desplazando tu plumilla electrónica —el cursor— a la palabra equivocada, y suprimiendo o insertando los caracteres adecuados.

Si las faltas de ortografía no son un problema para tí, pero en cambio te parece problemática la redacción de un texto, especialmente si se trata de una carta importante, tal como una instancia para solicitar un empleo o una carta al director de tu banco para explicarle las razones por las que inadvertidamente has incurrido en un descubierto de tu cuenta corriente, en tal caso este programa te resultará ideal. Al escribir una carta, puede ser que termines por gastar una gran cantidad de hojas en tu esfuerzo por encontrar las palabras adecuadas; normalmente las primeras frases suelen ser las que más dificultades presentan. Si realizas la composición directamente sobre el teclado de tu ordenador, mantendrás tu papelera vacía y tu humor intacto. En vez de pasar penas esforzándote en encontrar la forma correcta de empezar, puedes limitarte simplemente a hacer un borrador de tu carta. Después puedes analizar lo que has escrito, examinando el texto hacia arriba y hacia abajo. Una vez que hayas decidido lo que tienes que corregir o suprimir puedes borrar e insertar palabras o frases hasta que finalmente llegues a tener una composición perfecta.

Una ventaja adicional de componer directamente en pantalla es que puedes archivar la carta para usarla otra vez, o simplemente para tener una copia de lo que en ella decías.

Las utilidades contenidas dentro de este programa son análogas, aunque bastante menos sofisticadas, a las de las modernas máquinas de fotocomposición que se utilizan para la producción de publicaciones como **INPUT**. Si conoces los métodos de composición utilizados antes de que se inventara la máquina de fotocomposición, te resultará fácil apreciar las ventajas que te ofrece un paquete de utilidades de este tipo.

El procedimiento de utilización de este programa sigue las líneas generales anteriormente citadas. Las utilidades de edición en pantalla que trae el **Commodore 64** son muy eficaces y se conservan en este editor de textos. Pero cuando pases al modo de edición, los controles de edición pasarán en gran parte a las teclas de funciones, algunas de las cuales se utilizan para las funciones extra asociadas con la rutina de impresora.

La creación y modificación de texto tiene lugar en el área de trabajo pró-



ximo al fondo de la pantalla. En la parte superior de la pantalla se visualiza el texto que permanece almacenado en la memoria. El borrado e inserción de texto se hace de la forma acostumbrada utilizando las teclas INST SHIFT o DEL, después de situar el cursor con las teclas de CRSR a izquierda y derecha. La escritura se producirá sobre los caracteres ya existentes a menos que utilices el modo de inserción, pulsando la tecla f5.

El almacenamiento del texto en la memoria y su presentación en la parte superior de la pantalla se produce pulsando la tecla RETURN. En cualquier momento puedes acceder a dicho texto introduciendo el modo «editor» mediante pulsación de la tecla f1. La marca * indica el punto de texto a que tienes acceso en cada momento. Para transferir al área de trabajo la línea de texto que está encima



de la marca, pulsa la tecla f3 ; a continuación ya puedes editar de la forma acostumbrada.

Para borrar una línea de texto que esté almacenada en la memoria, pasa al modo editor, sitúa la marca inmediatamente por encima de la línea que quieres suprimir y pulsa la tecla INST/DEL. Si quieres tener una línea en blanco, pulsa simultáneamente INST y SHIFT, o bien pulsa RETURN sin introducir texto alguno en la zona de trabajo.

En el modo de edición, puedes visualizar el contenido de la memoria pulsando f1, seguido de CRSR arriba o abajo con o sin SHIFT para moverte en una u otra dirección desde la posición de la marca a la que siempre vuelve el modo editor.

Puedes saltar hacia arriba de cinco en cinco líneas pulsando la tecla de flecha hacia arriba, o hacia abajo de cinco en cinco líneas, con la tecla de desplazamiento hacia la izquierda.

Para salir de los modos editor o de introducción (sucesivamente), pulsa f7. El texto que está almacenado en la memoria no se verá afectado por ello.

Teclea para Commodore-64

```
1 REM**[]SIGNIFICA ESPACIO
2 REM**ESTE TRATAMIENTO DE
  TEXTO NO TIENE COMO
3 REM**OBJETIVO SUSTITUIR A
  PROGRAMAS COMERCIALES
10 PRINT"[SHIFT+CLR/HOME]
   [CTRL+8]";CHR$(8)
20 POKE 53280,6:POKE 53281,0
30 DIMTX$(501):OW=0:EM=0:
   DN=4
35 GC$="[CLR/HOME]
   [23*CRSR abajo]"
40 BL$=CHR$(160):TL=1:CP=1:
   MW=80:TW=60:PL=60:TH=40:
   GP=10:LF$=CHR$(13)+
   CHR$(13)
42 LF$=LF$+CHR$(13):GOSUB
   5000
50 TX$(0)="[CTRL+4][CTRL+9]"
   :SW$="":FOR F=0TO39:TX$
   (0)=TX$(0)+"+":SW$=SW$+
   "[COMM+Y]":NEXTF
55 FOR Z=1 TO 40:SP$=SP$+" "
   :NEXT
60 TX$(0)=TX$(0)+"[CTRL+0]
   [CTRL+8]":TX$(1)="[CTRL+9]
   [CTRL+4][9*COMM+@].
   [9*COMM+@].[9*COMM+@].
   [9*COMM+@].[CTRL+0]
   [CTRL+8]"
70 PRINTCHR$(142)
71 PRINT"[SHIFT+CLR/HOME]
   [CRSR abajo][CTRL+9]"TAB
   (15);"[CTRL+4]MENU PRIN
   CIPAL[CTRL+8]":PRINT
   "[2*CRSR abajo]";TAB(14);
   "[CTRL+9]L[CTRL+0]OAD
   [CRSR abajo]"
72 PRINTTAB(14)"[CTRL+9]S
   [CTRL+0]AVE[CRSR abajo]":
   PRINTTAB(14)"[CTRL+9]C
   [CTRL+0]AMBIO E/S
   [CRSR abajo]"
80 PRINTTAB(14)"[CTRL+9]E
   [CTRL+0]DITAR[CRSR abajo]"
   :PRINTTAB(14)"[CTRL+9]B
   [CTRL+0]ORRAR MEMORIA
   [CRSR abajo]"
81 PRINTTAB(14)"[CTRL+9]I
   [CTRL+0]MPRIMIR
   [CRSR abajo]"
82 PRINTTAB(14)"[CTRL+9]A
   [CTRL+0]LTERAR IMPRESORA
   [CRSR abajo]":PRINT
   TAB(14)"[CTRL+9]F[CTRL+0]
   IN DEL PROGRAMA
   [CRSR abajo]"
85 PRINTTAB(12)"[CRSR abajo]
   [CTRL+9][CTRL+4][][]
   SELECCIONA UNA OPCION[][]
   [CTRL+8]"
90 GET B$:IFB$=""THEN 90
100 B=0:FORF=1TO8:IF B$=MID$
    ("LSCEBIAF",F,1)THEN B=F
102 NEXT F:IF B=0 THEN 90
110 ON B GOSUB 4500,4000,
    5000,1000,160,3000,5500,
    130
120 GOTO 70
130 PRINT QD$;TAB(15);
    "[CRSR arriba][CTRL+9]
    [CTRL+4]ESTAS SEGURO
    (S/N)?"
140 GET R$:IF R$<>"S"AND R$
    <>"N" THEN 140
150 IF R$="S"THEN PRINT
    "[SHIFT+CLR/HOME]":END
155 RETURN
160 PRINT"[SHIFT+CLR/HOME]
```

```
    [CRSR abajo][CTRL+9]";
    TAB(13);"[CTRL+4]BORRAR
    MEMORIA[CTRL+8]":PRINT
    TAB(10)"[CRSR abajo]
    [CRSR abajo]ESTAS SEGURO
    (S/N)?"
170 GET B$:IF B$<>"S" AND B$
    <>"N" THEN 170
180 IF B$="N" THEN RETURN
190 IF TL=1 THEN RETURN
195 FOR K=1 TO TL-1:TX$(K)=
    "":NEXT:TX$(1)=TX$(TL):
    TX$="":TL=1:CP=1:RETURN
1000 PRINT CHR$(14)
     "[SHIFT+CLR/HOME]
     [CTRL+N][CRSR abajo]
     [CTRL+9]";TAB(15);"
     [CTRL+4][SHIFT+E]DITAR
     [SHIFT+T]EXTO[CTRL+8]"
1001 PRINT"[CRSR abajo]
     [CRSR abajo]"TAB(14);
     "[CTRL+9][SHIFT+C]
     [CTRL+O]OMIENZO DEL
     TEXTO"
1002 PRINT TAB(14);"
     [CRSR abajo][CTRL+9]
     [SHIFT+F][CTRL+O]INAL
     DEL TEXTO":PRINTTAB(14)
     "[CRSR abajo][CTRL+9]
     [SHIFT+S][CTRL+O]IGUI
     ENTE LINEA DE TEXTO"
1004 PRINT TAB(14)"
     [CRSR abajo][CTRL+9]
     [SHIFT+M][CTRL+O]ENU
     PRINCIPAL"
1010 GETB$:IF B$=""THEN
     1010
1020 B=0:FORF=1TO4:IF B$=
     MID$("CFSM",F,1)THEN
     B=F:GOTO 1030
1022 NEXT F:GOTO 1010
1030 ON B GOTO 1050,1060,
     1070,1080
1040 IF TL=<3 OR CP=1 THEN
     1070
1050 CP=1:GOTO 1070
1060 CP=TL
1070 PRINT"[SHIFT+CLR/HOME]"
     :GOSUB 2090:GOSUB 1500:
     GOTO 1000
1080 RETURN
1500 A$=" ":P=0
1505 IF EM=1 THEN 2500
1510 POKE 198,0
1515 PRINT LEFT$(GC$,23)A$;
1520 CH=PEEK(1904+P):GET T$
1521 IF LEN(A$)=81 OR T$=
     CHR$(13)THEN GOSUB 2000
     :GOTO 1515
1522 CH=(CH+128)AND 255:POKE
     1904+P,CH:CH=(CH+128)
     AND 255:POKE 1904+P,CH
1530 IFT$=""OR T$=CHR$(34)
     THEN 1520
1540 IF T$="[CRSR arriba]"
     OR T$="[CRSR abajo]"
     THEN 1520
1550 IF P<LEN(A$)-1 AND T$=
     CHR$(10) THEN A$=LEFT$
     (A$,P)+MID$(A$,P+2):
     GOTO 1600
1551 IF T$=CHR$(10) THEN
     1520
1555 IF T$=CHR$(136) THEN
     RETURN
1560 IF T$=CHR$(134) AND CP
     =1 THEN 1520
1561 IF T$=CHR$(134) THEN
     A$=TX$(CP-1)+" ":T$="":
     P=0:GOSUB2090:GOTO 1510
1562 IF T$=CHR$(135) THEN OW
     =ABS(1-OW):GOSUB 2090:
     GOTO 1510
1563 IF T$="[CLR/HOME]" THEN
     CP=1:GOSUB 2090
1564 IF T$=CHR$(133) THEN EM
     =1:PM=CP:GOSUB 2090:
     GOTO 1505
1565 IF T$="[SHIFT+CLR/HOME]
     " THEN GOSUB 2090:GOTO
     1500
1567 IF T$<>CHR$(20) THEN
     1580
1570 IF P=0 THEN 1510
1572 A$=LEFT$(A$,P-1)+MID$
     (A$,P+1):P=P-1
1575 PRINT LEFT$(GC$,23);:
     FOR F=1 TO LEN(A$)-2:
     PRINT "[CRSR dcha.]";:
```

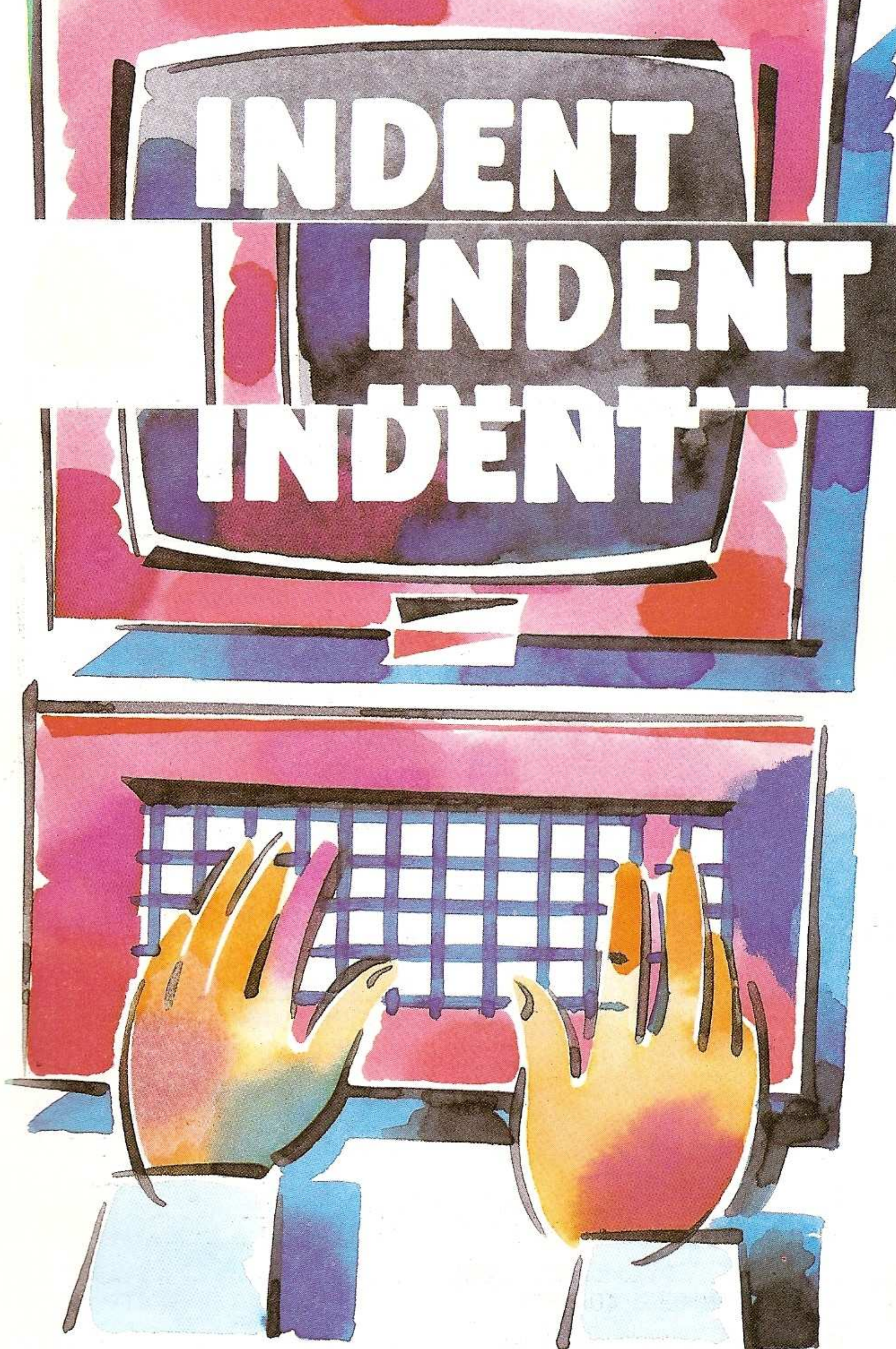


```
     NEXT
1577 PRINT"[][]":GOTO 1510
1580 KJ=0:IF T$=CHR$(148)
     THEN T$=" ":KJ=1
1582 IF T$<>CHR$(157) AND
     T$<>CHR$(29)AND ASC(T$)
     <32 THEN 1510
1590 IF T$=CHR$(29) OR T$=
     CHR$(157) THEN 1610
1591 IF OW=1 AND J<>1 THEN
     1595
1593 A$=LEFT$(A$,P)+T$+MID$
     (A$,P+1):P=P+1:GOTO
     1600
1595 B$=A$:A$=LEFT$(A$,P)+T$
     +MID$(A$,P+2)
1598 P=P+1:A$=A$+" "
1600 PRINT LEFT$(GC$,23)A$;
1610 IF T$=CHR$(29) AND DP<
     LEN(A$)-1 THEN P=P+1
1620 IF(T$=CHR$(157)AND P>0)
     OR KJ=1 THENLIST1630
```



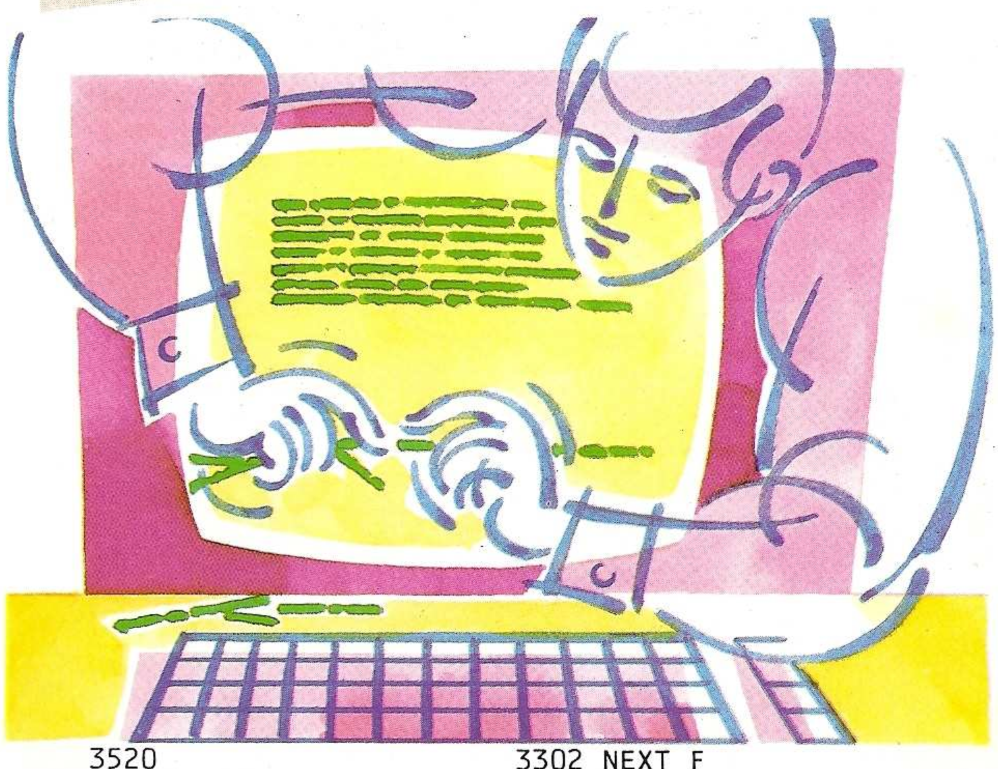
```
1630 GOTO 1520
1700 IF CP=1 THEN 1510
1710 CP=CP-1:FOR F=CP TO TL:
     TX$(F)=TX$(F+1):NEXT:
     TL=TL-1
1720 GOSUB 2090
1730 GOTO 1510
1800 IF TL>499 THEN 1510
1810 FOR F=TL+1 TO CP+1 STEP
     -1:TX$(F)=TX$(F-1):NEXT
     :TL=TL+1:TX$(CP)=""
1820 GOSUB 2090:GOTO 1510
2000 X=0:IF TL>499 THEN 2060
2001 IF LEN(A$)<41 THEN TT$
     (X)=LEFT$(A$,LEN(A$)-1)
     :A$="":GOTO 2030
2010 FOR I=41 TO 1 STEP-1:IF
     MID$(A$,I,1)<>" "THEN
     NEXT:I=41
2020 TT$(X)=LEFT$(A$,40):A$
     =MID$(A$,41)
2030 X=X+1:IF A$<>""AND A$<>
     " " THEN 2001
2040 FOR I=TL+X TO CP+X STEP
     -1:TX$(I)=TX$(I-X):
     NEXT I
2050 FOR I=0 TO X-1:TX$(CP+
     I)=TT$(I):NEXT I
2060 A$=" ":P=0:PRINT LEFT$
     (GC$,23)A$;
2080 TL=TL+X:CP=CP+X
2090 IF CP<15 THEN S1=0:GOTO
     2100
2095 S1=CP-15
2100 PRINT"[SHIFT+CLR/HOME]"
     ;:FOR K=S1 TO S1+15:
     PRINT TX$(K);:IF LEN
     (TX$(K))<40 THEN PRINT
     CHR$(160)
2110 IF K=CP-1 THEN PRINT
     "[COMM+7][CTRL+9][]
     [CTRL+0][CTRL+8]"
2120 NEXT
2122 PRINT"[CTRL+9][FLECHA
     izq.]"SP$"[CTRL+4]0
     [COMM+7]123456789
     [CTRL+4]0[COMM+7]123456
     789[CTRL+4]0[COMM+7]123
     456789[CTRL+4]0[COMM+7]
     123456789[CTRL+8]
     [CTRL+0]";
2130 PRINT"[CTRL+5]MEM LIBRE
     =[COMM+6]";40*(501-TL);
2140 PRINT"[COMM+6]MODO O/W=
     [COMM+6]"OW"[COMM+6]
     EDICION=";EM:PRINT
     "[CTRL+6]"SW$"[CTRL+8]"
     ;:RETURN
2500 IF CP<5 THEN PM=CP
2510 GET TB$:IF TB$=""THEN
     2510
2512 IF PM=1 AND TB$=
     "[CRSR arriba]" THEN
     2510
2520 IF PM>1 AND TB$=
     "[CRSR arriba]" THEN PM
     =PM-1:CP=PM:GOTO 2550
2525 IF PM<TL AND TB$=
     "[CRSR abajo]" THEN PM=
     PM+1:CP=PM:GOTO 2550
2530 IF PM>5 AND TB$="[FLECHA
     arriba" THEN PM=PM-5:CP=
     PM:GOTO 2550
2535 IF PM<TL-5 AND TB$=
     "[FLECHA izq.]" THEN PM=
     PM+5:CP=PM:GOTO 2550
2540 GOTO 2560
2550 GOSUB 2090:GOTO 2510
2560 IF TB$=CHR$(136) THEN EM
     =0:GOSUB 2090:GOTO 1505
2570 IF TB$<>CHR$(148) THEN
     2580
2571 IF CP<1 THEN 2510
2572 FOR K=TL+1 TO PM+1 STEP
     -1:TX$=TX$(K-1):NEXT:TL
     =TL+1:TX$(PM)=""
2573 GOSUB 2090:GOTO 2510
2580 IF TB$<>CHR$(20) THEN
     2590
2581 IF PM=TL THEN 2510
2582 FOR K=PM TO TL:TX$(K)=
     TX$(K+1):NEXT:TL=TL-1
2583 TX$(TL+1)="":GOSUB 2090
     :GOTO 2590
2584 CP=CP-1
2590 IF TB$="@"THEN SF=SF+1:
     IF SF=1 THEN SS=CP:
     GOTO 2510
2600 IF SF=2 THEN SE=CP:SF=0
     :GOSUB 5130:GOTO 2510
2610 IF TB$="S"THEN GOSUB
     5070
2620 GOTO 2510
3000 IF TL<2 THEN 3050
3010 PRINT"[SHIFT+CLR/HOME]
     [CRSR abajo][CTRL+9]"
     TAB(12)"[CTRL+4]RUTINA
     DE IMPRESORA[CTRL+8]":
     CLOSE4
3020 PRINT"[2*CRSR abajo]
```

```
     [CRSR dcha.]DESDE
     [CTRL+9]M[CTRL+0]EMORIA
     O DESDE [CTRL+9]F
     [CTRL+0]ICHERO?"
3030 GET R$:IF R$<>"M" AND
     R$<>"F" THEN 3030
3040 IF R$="M" THEN 3060
3050 GOSUB 4500
3060 IF TL=1 THEN PRINT TAB
     (11)"[3*CRSR abajo]
     [CTRL+9][CTRL+4]NO HAY
     FICHERO EN MEMORIA
     [CTRL+8]":GOTO 3570
3070 KF=0:PRINT "[CRSR dcha.]
     [CRSR abajo]LLENA
     BLOQUES VARIABLES
     (S/N)?"
3080 GET R$:IF R$<>"S" AND
     R$<>"N" THEN 3080
3090 IF R$="N" THEN 3150
3100 PRINT:PRINT "[CRSR
     abajo][CRSR dcha.]
     [CTRL+9]T[CTRL+0]ECLADO
     O [CTRL+9]F[CTRL+0]
     ICHERO?"
3110 GET R$:IF R$<>"T" AND
     R$<>"F" THEN 3110
3120 K=2:IF R$="T" THEN KF=1
     :GOTO 3150
```




```
3130 INPUT"[CRSR abajo]
     ESCRIBE NOMBRE DEL
     FICHERO";VB$:VB$=LEFT$
     (VB$,16)
3140 IF LEFT$(VB$,1)<A OR
     LEFT$(VB$,1)>"Z" THEN
     3130
3150 PRINT"[SHIFT+CLR/HOME]
     QUIERES CAMBIAR LA
     IMPRESORA(S/N)?"
3160 GET R$:IF R$<>"S" AND
     R$<>"N" THEN 3160
3170 IF R$="S" THEN GOSUB
     3160
3180 PRINT"[SHIFT+CLR/HOME]"
3190 VB=0:PP=0:AS=0:LC=1:
     PRINT"QUIERES VER LA
     SALIDA POR PANTALLA";
3192 PRINT(S/N).<RETURN>
     PARA VOLVER AL MENU
     PRINCIPAL":BL=0
3200 GET R$:IF R$<>"S" AND
     R$<>"N" AND RZT$<>CHR$
     (13) THEN 3200
3210 IF R$=CHR$(13) THEN
     RETURN
3211 IF R$="N" THEN P=DN:
     OPEN4,P,7,"[CRSR abajo]
     ":GOTO 3220
3215 OPEN 3,4:PRINT CHR$(14)
3220 IF KF<>2 THEN 3240
```

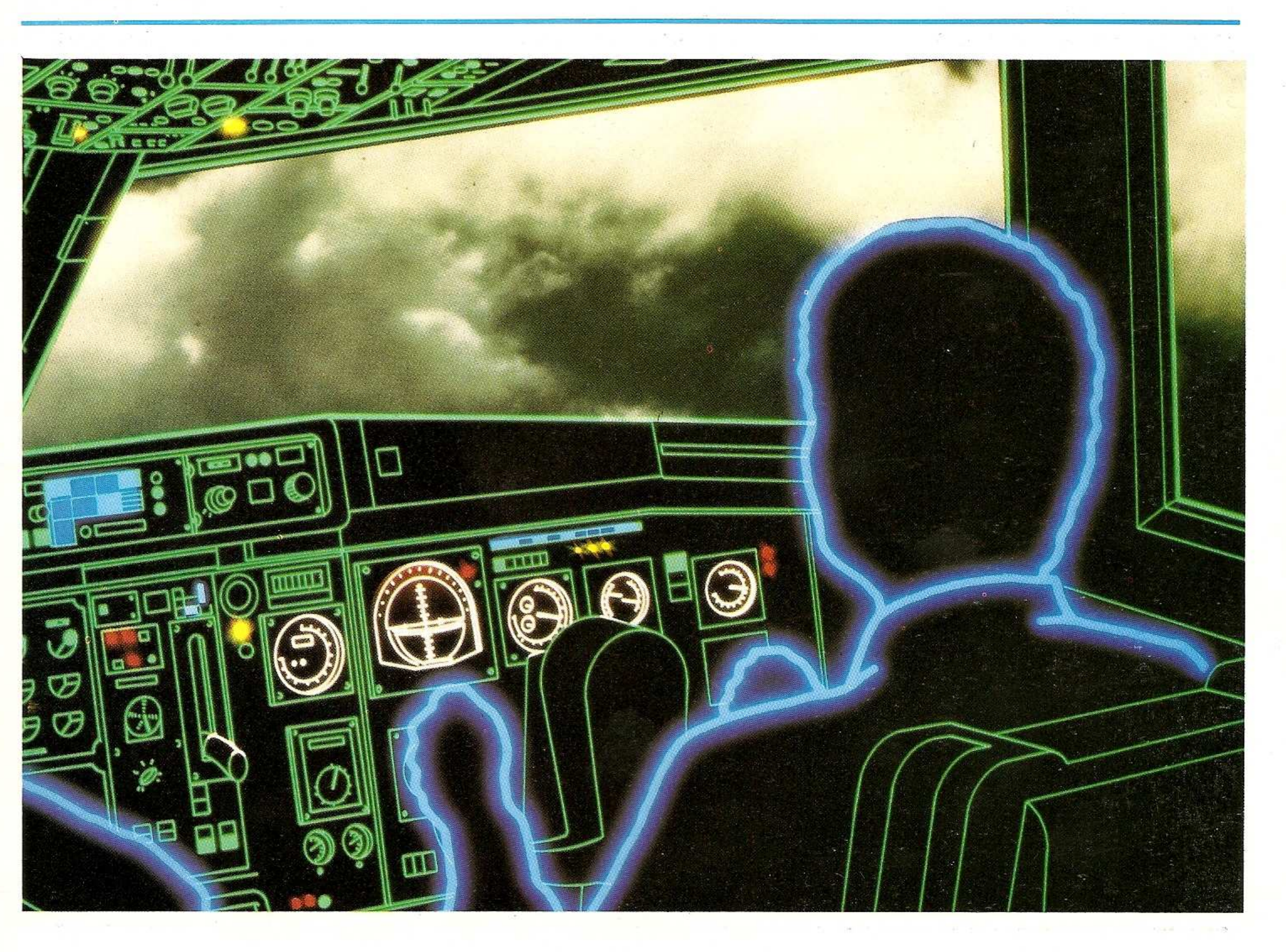
```
3230 IF DL=1 AND KF=2 THEN
     3232
3231 IF KF=2 THEN 3236
3232 OPEN 2,8,2,VB$+"S,R":
     INPUT#2,DV,DV:GOTO 3240
3236 OPEN 1,1,0:INPUT#1,
     DV,DV
3240 GP$="":IF R$="N" THEN
     FOR F=1 TO GP:GP$=GP$+
     " ":NEXT
3250 FOR K=1 TO L-1
3252 IF LEFT$(TX$(K),1)="#"
     AND LEN(TX$(K))-1>AS
     THEN AS=LEN(TX$(K))
3260 NEXT:IF AS>TW THEN
     PRINT "ERROR:DIRECCION
     MUY LARGA":GOTO 3570
3270 K=1:PRINT#4,LF$;GP$;:
     AS$="":IFAS>0THEN FOR F
     =1 TO GP+TW-AS:AS$=AS$+
     " ":NEXT
3280 TT$=TX$(K)
3290 IF T$=""THEN PRINT#4,
     CHR$(13);GP$;:PP=0:LC=
     LC+1:GOSUB 3590:GOTO
     3520
3300 BP=0:FOR F=1 TO LEN
     (TT$):IF MID$(TT$,F,2)=
     "][" THEN BP=F:GOTO 3304
3302 NEXT F
3304 IF BP=0 OR KF=0 THEN
     3390
3310 IF KF=1 THEN 3370
```



```
3320 IF DL=1 THEN 3360
3330 IF ST=64 THEN 3350
3340 INPUT#1,RP$:GOTO 3380
3350 PRINT"ERROR-NO HAY SUFI
     CIENTES DATOS EN EL
     FICHERO":GOTO 3570
3360 IF ST=64 THEN 3350
3362 INPUT#2,RP$:GOTO 3380
3370 BL=BL+1: PRINT:RP$="":
     PRINT"ESCRIBE BLOQUE
     VARIABLES";BL;:INPUT
     RP$
3380 TT$=LEFT$(TT$,BP-1)+RP$
     +MID$(TT$,BP+2):GOTO
     3300
3390 CF=0:FOR F=1 TO 4:IF
     LEFT$(TT$,1)=MID$("&$*#
     ",F,1) THEN CF=F
3391 NEXT F:ON CF GOTO 3460,
     3470,3490,3510
3400 IF PP+LEN(TT$)<=TW THEN
     PRINT#4,TT$;:PP=PP+LEN
     (TT$):GOTO 3520
3410 TA$=LEFT$(TT$,TW-PP)
3420 CF=0:FOR F=1 TO LEN(TT$)
     :IF MID$(TT$,F1)=" "
     THEN CF=F:GOTO 3422
3421 NEXT F:IF CF>TW THEN
     PRINT"ERROR-PALABRA MUY
     LARGA EN ";TT$:GOTO 3570
3430 IF RIGHT$(TA$,1)=" "
     THEN 3450
3440 IF LEN(TA$>0 THEN TA$=
     LEFT$(TA$,LEN(TA$)-1):
     GOTO 3430
3450 PRINT#4,TA$;CHR$(13);
     GP$;:PP=0:LC=LC+1:GOSUB
     3590:TT$=MID$(TT$,LEN
     (TA$)+1)
3452 IF TT$<>""THEN BP=1:
     GOTO 3400
3454 GOTO 3520
3460 PRINT#4,CHR$(13);CHR$
     (13);GP$;:PP=0:LC=LC+1:
     GOSUB 3590:TT$=MID$(TT$
     ,2):GOTO 3300
3465 GOTO 3300
3470 TT$=MID$(TT$,2):PRINT#4
     ,CHR$(13);GP$;:IF PP<>
     TW THEN 3479
3471 FOR F=1 TO INT(TX/2):
     GOTO 3480
3479 PP=0
3480 LC=LC+1:GOSUB 3590:GOTO
     3300
3490 TT$=MID$(TT$,2):IF LEN
     (TT$)<=TW THEN 3500
3491 PRINT"ERROR-NO SE PUEDE
     CENTRAR"TT$:GOTO 3520
3500 PRINT#4,CHR$(13);GP$;:
     FOR F=1 TO INT((TW-LEN
     (TT$))/2):PRINT#," ";:
     NEXT
3501 PRINT#4,TT$
3506 PRINT#4,CHR$(13);GP$;:
     PP=0:LC=LC+1:GOSUB 3590
     :GOTO 3520
3510 PRINT#4,CHR$(13);AS$;
     MID$(TT$,2);:PP=0:LC=LC
     +1:GOSUB 3590
3520 K=K+1:IF P=3 THEN FOR Z
     =1 TO 500:NEXT
3530 IF K<TL THEN 3280
3540 IF P<>3 THEN PRINT#4,
     LF$;LF$:GOTO 3550
3541 PRINT:PRINT
3550 FOR Z=1 TO 4:CLOSE Z:
     NEXT Z
3560 IF P=0 THEN 3190
3561 RETURN
3570 FOR Z=1 TO 3000:NEXT:
     FOR Z=1 TO 8:CLOSE Z:
     NEXT Z
3580 RETURN
3590 IF C>TH THEN PRINT#4,
     LF$;LF$;GP$;:LC=1
3600 RETURN
5000 RETURN
```

LA APROXIMACION FINAL

- RUTINA DE CONTROL DEL TECLADO
- COMPROBACION DE ATERRIZAJE
- SOBRE LA PISTA
- CHOQUES
- ESTIMACION DE DAÑOS



Quita el piloto automático, pero no exhales todavía el suspiro de alivio: aún tienes los mandos. Utilizando únicamente seis teclas debes conseguir que el avión se pose suavemente sobre la tierra

Con esta tercera y última parte del programa simulador de vuelo tendrás por fin el control del aparato.

Hay tres pares de teclas que te permiten aumentar y disminuir la velocidad del motor, hacer que el avión vire o que suba y baje.

Te encontrarás situado a 20000 metros de la pista de aterrizaje y a 2000 metros de altura. No es fácil manejar un avión, en especial si eres un piloto inexperto. No esperes convertirte en experto con unos cuantos vuelos, la capacidad de pilotar un avión es una habilidad que muy pocos adquieren en poco tiempo y con facilidad.

Cuando vayas perseverando y adquiriendo más experiencia en el control de la aeronave, tus maniobras de aterrizaje irán mejorando. Después de cada aterrizaje (o cada accidente) recibirás un informe con los detalles finales del vuelo. Estúdialo para ver qué es lo que has hecho mal, e intenta corregir las deficiencias.

El simulador de vuelo sólo necesita unas cuantas líneas más para estar completo. No te olvides de utilizar el cartucho del **Simon's BASIC**.

Teclea para C-64

```
3000 GET K$:IF K$=""THEN
     RETURN
3010 IF K$="S" AND TC>.2
     THEN TC=TC-.2
3020 IF K$="F" AND TC<8.8
     THEN TC=TC+.2
3030 IF K$="Q" THEN PT=PT+1
3040 IF K$="A" THEN PT=PT-1
3050 IF K$="O" AND RL>-30
     THEN RL=RL-1
```

```
3060 IF K$="P" AND RL<30
     THEN RL=RL+1
3070 RETURN
5500 GOSUB 2000:GOSUB 3000:
     GOSUB 1000
5510 IF PZ<=0 THEN 6000
5520 GOSUB 2000:GOSUB 2060
5530 GOTO 5500
6000 IF ABS(RL)>RT OR PT>TP
     OR PT<0 OR AS>80 THEN
     6100
6010 IF ABS(PX)>WR OR ABS(PY
     )>1000 THEN 6200
6020 PAUSE 1:PRINT"[SHIFT+
     CLR/HOME]FELICIDADES
     POR UN BUEN ATERRIZAJE"
     :GOTO 6500
6100 FOR Z=1 TO 15:LINE 80,
     55,RND(1)*160,RND(1)*
     110,RND(1)*3+1:COLOUR
     6,Z
6110 NEXT Z:PAUSE 3
6120 IF AS<40 THEN PRINT
     "[SHIFT+CLR/HOME]UN
     GOLPE COMO ESE HAS
     DESTROZADO EL"
6130 PRINT"AVION Y HAS MA
     TADO A LOS PASAJEROS":
     GOTO 6500
6200 PAUSE 1:PRINT"[SHIFT+
     CLR/HOME]TE SALISTE DE
     LA PISTA"
6210 IF AS<40 THEN PRINT
     "AFORTUNADAMENTE NO
     IVAS TAN DEPRISA"
6215 IF AS<40 THEN PRINT
     "COMO PARA CAUSAR MU
     CHOS DA#OS" :GOTO 6500
6220 IF AS<80 THEN PRINT"A
     ESA VELOCIDAD SALES CON
     POCOS DA#OS"
6225 IF AS<80 THEN PRINT"Y
     LIGERAS CONTUSIONES"
     :GOTO 6500
6230 PRINT"SALIR DE LA PISTA
     A TAL VELOCIDAD NO DEJA
     SUPERVIVIENTES!"
6500 COLOUR 6,6:NRM:PRINT
     "[5*CRSR abajo][4*CRSR
     dcha.]DETALLES FINALES
     DEL VUELO"
6510 PRINT"[CRSR abajo]VELO
```



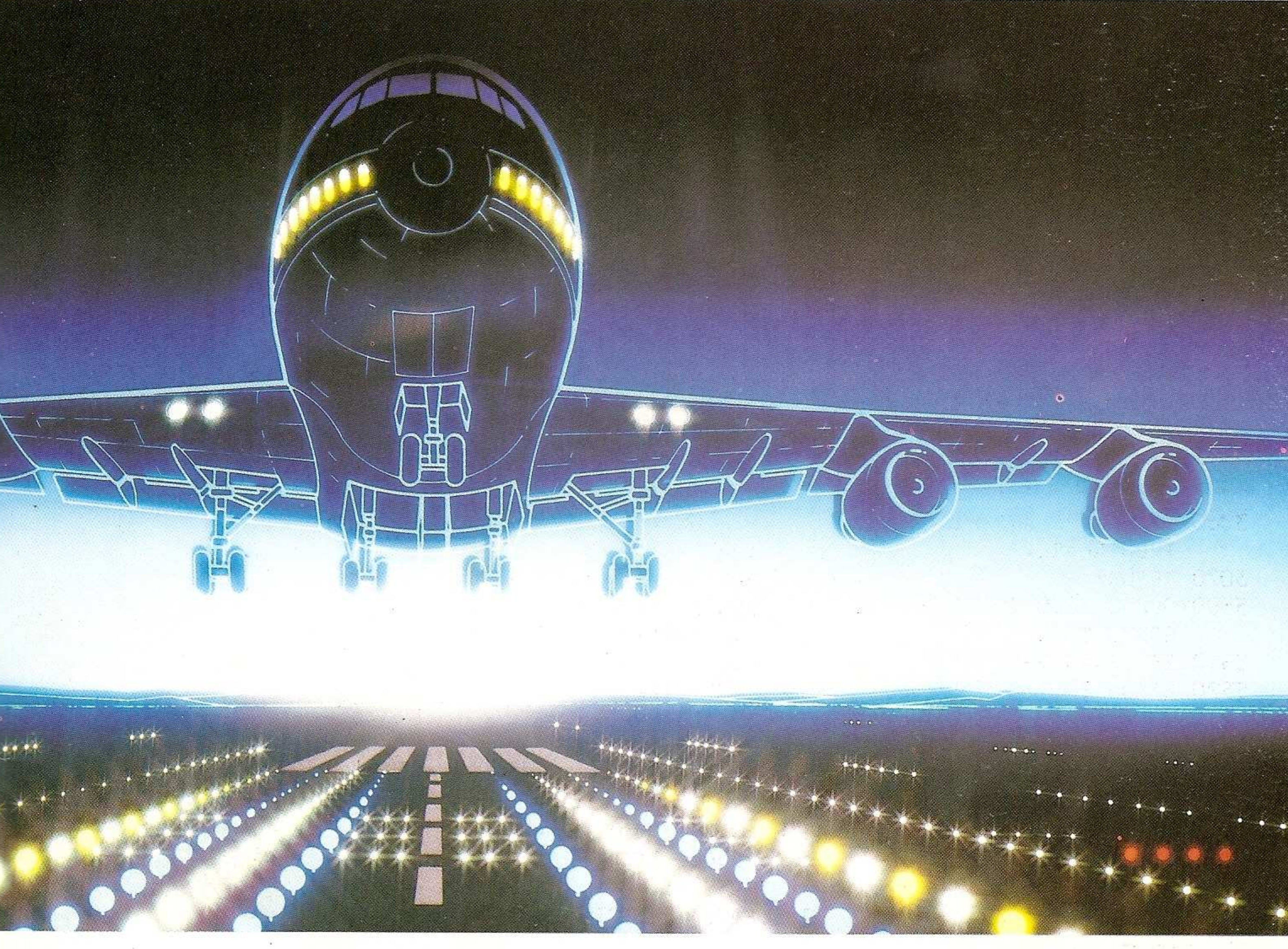
```
      CIDAD DEL AIRE=";INT
      (AS);"M/S"
6515 PRINT"DISTANCIA =";INT
      (SQR(PY*PY+PX*PX)):
      PRINT"[3*CRSR dcha.]
      CABECEO =";PT
6520 PRINT"[4*CRSR dcha.]
      BALANCEO =";RL:PRINT
      "[5*CRSR dcha.]RPM =";
      INT(10*TC)/10;"X 1000"
6530 PRINT"[3*CRSR dcha.]
      DERIVA =";INT(ABS(PX));
      "METROS":PRINT"[CRSR
      dcha.]ORIENTACION =";AD
      ;"GRADOS"
6540 PRINT"[2*CRSR abajo]
      OTRA VEZ ([CTRL+6]S/N
      [CTRL+4)?":FLASH 5,10:
      POKE 650,0
6550 GET A$:IF A$<>"S" AND
      A$<>"N" THEN 6550
6560 OFF:IF A$="N" THEN
      PRINT"[SHIFT+CLR/HOME]
      [CTRL+1":COLOUR 6,1:END
6570 RUN
```

El bucle principal que se extiende entre las líneas 5500 y 5530, ha sido modificado de forma que se pueda utilizar la subrutina de control.

Las líneas 3000 a 3070 confieren al teclado el control del aeroplano. Las líneas 3010 y 3020 te permiten utilizar [S] o [F] para hacer que el avión vaya más lento o más rápido. Lo que hacen [S] y [F] es modificar el valor del régimen de revoluciones del motor, incrementando o decrementando la variable TC. De forma análoga, [Q] y [A] alteran la inclinación del avión, en otras palabras, al presionar [Q], el avión tenderá a descender, mientras que si aprietas [A] tenderá a elevarse, incrementando o decrementando PT. Análogamente, las líneas 3050 y 3060 hacen que [O] y [P] actúen modificando el balanceo de la aeronave, es decir, te permiten hacer virajes.

Si en la línea 5510 PZ resulta ser igual o menor que cero, el avión ha tocado el suelo y el programa salta a la línea 6000. Las líneas 6000 a 6540 se ocupan de determinar dónde has tomado tierra y de comprobar si lo has hecho bien o mal.

La línea 6000 comprueba si el balance y la inclinación están dentro de los límites permitidos, de si el rumbo seguido es el correcto y de si la velocidad del aire está por debajo de los 80 metros por segundo. Si la velocidad o el rumbo seguido por el avión están fuera de los límites, el programa salta a la línea 6100, que dibuja una serie de grietas en la ventanilla de la cabina. Después de una corta PAUSE, las



líneas 6120 y 6130 abordan una investigación y un posible proceso por parte de las autoridades de Aviación Civil; naturalmente ello significa que te has estrellado.

La línea 6010 comprueba la posición del aeroplano con relación a la pista de aterrizaje. La línea 6200 se ocupa de presentar las malas noticias: ATERRIZAJE FUERA DE LA PISTA. Si la velocidad del aire en el momento de tomar tierra era menor que 40 metros por segundo, las líneas 6210 y 6215 te dicen al piloto lo siguiente: AFORTUNADAMENTE NO IBAS DEMASIADO RAPIDO Y LOS DAÑOS NO SON MUCHOS. El programa salta entonces a la línea 6500.

Si la velocidad del aire estaba comprendida entre 40 y 80 metros por segundo, aparece el mensaje: CON ESTA VELOCIDAD HAS PODIDO ESCAPAR CON LIGEROS DAÑOS Y UNOS CUANTOS ARAÑAZOS. Si la velocidad era mayor que 100 metros por segundo, se trata de un caso para las autoridades de Aviación Civil...

Cuando el avión aterriza correctamente, el ordenador salta a la línea 6020, que se encarga de enviar el mensaje de aterrizaje con éxito. Cualquiera que sea el resultado del aterrizaje, el siguiente paso es la rutina que contiene los detalles del vuelo, la cual comienza en la línea 6500.

La línea 6500 presenta el título de los detalles, mientras que las líneas 6510 a 6530 presentan todos los valores de las variables asociadas con el avión. Esta información es la que te permite juzgar exactamente lo bien que lo has hecho.

Al final del programa figura una rutina de ¿OTRA VEZ? por si quieres intentarlo de nuevo.

Observa que las agujas de los relojes mantienen las referencias de las condiciones iniciales del vuelo a modo de comparación para tu mejor control.

LAS SERPIENTES SUMADORAS

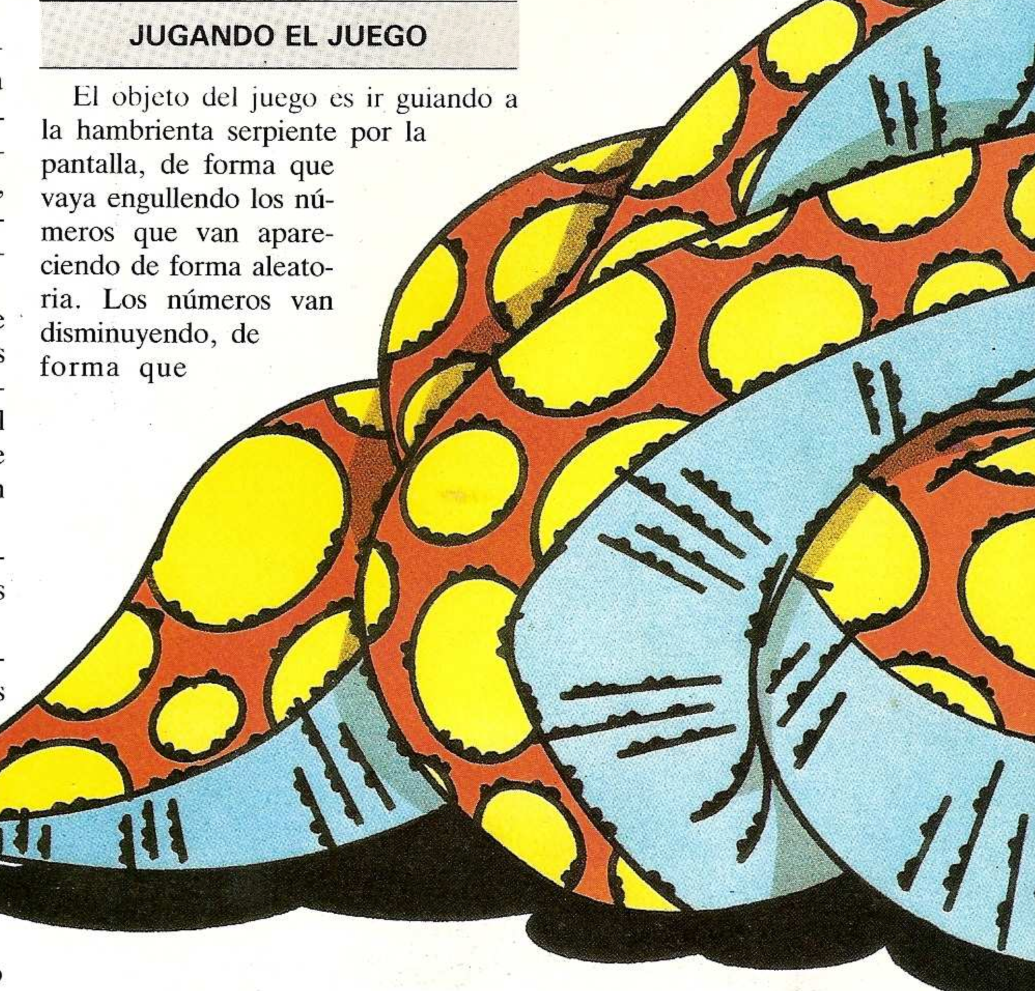
Guía a la serpiente hambrienta para que coma en el «Juego de la serpiente de INPUT». Al engullir ávidamente los suculentos números, la pequeña serpiente irá creciendo hasta hacerse enorme. Naturalmente, siempre que tengas la necesaria destreza.

El juego de la serpiente es uno de los más conocidos y más fáciles de jugar, pese a lo cual continúa siendo enormemente enviciante. Por suerte, no hace falta recurrir al código máquina para programar un juego de este tipo; precisamente este es un juego que ha marcado época en la historia de los ordenadores domésticos, figurando como uno de los más satisfactorios que pueden escribirse en BASIC.

JUGANDO EL JUEGO

El objeto del juego es ir guiando a la hambrienta serpiente por la pantalla, de forma que vaya engullendo los números que van apareciendo de forma aleatoria. Los números van disminuyendo, de forma que cuanto más tarde la serpiente, más baja será la puntuación. Si tardas demasiado y el número se decrementa hasta cero, desaparecerá y en su lugar aparecerá otro número. Cada vez que la serpiente se traga un número, su longitud aumenta en el correspondiente número de segmentos.

Ten cuidado de no sobrepasar los bordes y de no permitir que la serpiente se entrecruce consigo misma, cosa que te irá resultando más difícil a medida que se va haciendo más larga. Si





- EL CLASICO JUEGO EN UNA VERSION PARA BASIC
- DIBUJO DE LA COMIDA
- COMIENDO NUMEROS
- LA CULEBRA SE EXTIENDE

atraviesas uno de los bordes o el cuerpo de la serpiente, el juego terminará y aparecerá un mensaje de FUERA que cubre toda la pantalla.

Teclea para C-64

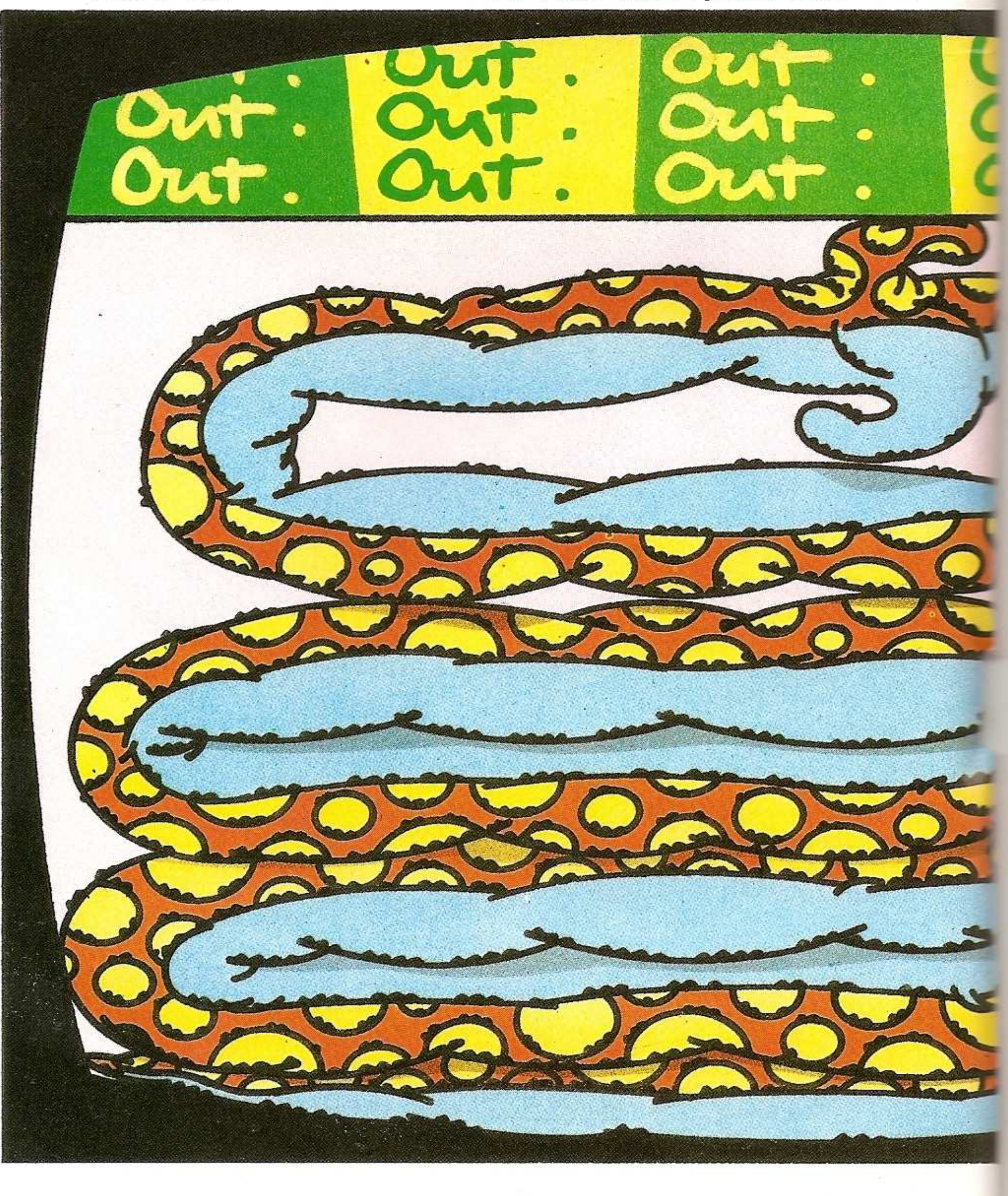
```
10 POKE 53280,2:POKE 53281,1
   :PRINT "[SHIFT+CLR/HOME]"
20 HS=0
30 DIM S(1000,2)
40 POKE 54296,0
100 S=0
110 S(1,1)=10:S(1,2)=14
120 S(2,1)=11:S(2,2)=14
130 S(3,1)=12:S(3,2)=14
135 GOSUB 1500
140 T=1:H=3
145 YV=1:XV=0
150 FOR N=1 TO 3:SP=1024+S
    (N,1)*40+S(N,2):POKESP,
    163:POKE 54272+SP,6:NEXT
160 Y=12:X=14
165 P=0
170 GOSUB 1000
190 SP=PEEK(1024+Y*40+X):IF
    SP=160 OR SP=163 OR Y=0
    THEN 2000
200 H=H+1:IF H=1000 THEN H=1
210 SP=1024+Y*40+X:POKE SP,
    163:POKE SP+54272,6
220 S(H,1)=Y:S(H,2)=X
230 SP=S(T,1)*40+S(T,2)+1024
    :POKE SP,32
240 T=T+1:IF T=1000 THEN T=1
245 IF P<>0 THEN 260
250 P=INT(RND(1)*9)+1:FY=INT
    (RND(1)*21)+2:FX=INT(RND
    (1)*37)+1
255 IF PEEK(1024+FY*40+FX)<>
    32 THEN 250
260 SP=1024+FY*40+FX:
    POKE SP
    ,P+48:POKE
    SP+54272,0
270 IF RND(1)<.95 THEN 290
280 P=P-1:IF P=0 THEN POKE
    1024+FX+FY*40,32
```

```
290 IF FY<>Y OR FX<>X THEN
    170
300 S=S+P:SP=1024+Y*40+X:
    POKE SP,163:POKE SP+
    54272,6:PRINT
    "[CLR/HOME][CTRL+9]
    [6*CRSR dcha.]"S
310 FOR N=1 TO P
320 GOSUB 1000
325 SP=PEEK(1024+Y*40+X):IF
    SP=160 OR SP=163 OR Y=0
    THEN 2000
330 H=H+1:IF H=1000 THEN
    H=1
340 S(H,1)=Y:S(H,2)=X
350 SP=1024+S(H,1)*40+S(H,2)
    :POKE SP,163:POKE SP+
    54272,6
355 FOR M=1 TO 10: NEXT
360 NEXT N
500 GOTO 165
1000 GET A$
1010 IF A$="Q" THEN YV=-1:
     XV=0
1020 IF A$="A" THEN YV=1:
     XV=0
1030 IF A$="O" THEN XV=-1:
     YV=0
1040 IF A$="P" THEN XV=1:
     YV=0
1050 Y=Y+YV:X=X+XV:RETURN
1500 FORN=0TO39:POKE1024+N,
     160:POKE55296+N,0:POKE
     1984+N,160:POKE56256+N
     ,0:NEXT
1510 FOR N=1TO23:POKE1024+N
     *40,160:POKE 1063+N*40
     ,160
1515 POKE 55296+N*40,0:POKE
     55335+N*40,0:NEXT
1520 PRINT"[CLR/HOME][CTRL+1]
     [CTRL+9]PUNTOS"S;TAB(18)
     "[CTRL+9]MAYOR
     PUNTUACION";HS
1590 RETURN
2000 FOR F=0 TO 24: POKE
     54272+F,0:NEXT
2010 POKE 54273,8
2020 POKE54277,175:POKE54278,
     30:POKE54296,15:POKE
     54276,129
2030 PRINT"[CLR/HOME]";TAB
     (11)"[CTRL+3][CTRL+9]
     FUERA!"
2040 FOR F=0 TO 10:PRINT
     "[CLR/HOME]"TAB(11)
     "FUERA!":FOR G=0 TO 90:
     NEXT:PRINT"[CLR/HOME]"
     TAB(11)"[CTRL+9]FUERA"
2050 FOR G=0 TO 90:NEXT G,F:
     POKE 54277,0:POKE
     54278,0
2060 IF S>HS THEN HS=S
2070 PRINT"[SHIFT+CLR/HOME]
     [8*CRSR abajo]"TAB(15)
     "PUNTOS"S:PRINT"[2*CRSR
     abajo]"TAB(13)"MAYOR
     PUNTUACION"HS
2080 PRINT TAB(3)"[CTRL+9]
     [2*CRSR abajo]PULSA UNA
     TECLA PARA JUGAR DE
     NUEVO."
2090 POKE 198,0:WAIT 198,1:
     PRINT"[SHIFT+CLR/HOME]"
     :GOTO 100
```

El primer listado del programa principal está hecho para el **Commodore 64**, pero si dispones de un **Vic** con una ampliación de memoria RAM de 3K, puedes jugar el juego, modificando las líneas que figuran a continuación y no utilizando las líneas 2010 y 2020. Hemos tenido que limitar el tamaño de la zona de juego a causa de lo limitado de la memoria.

Teclea para Vic-20

```
1 REM**EN EL VIC CON
  33K DE EXPANSION SE TECLEA
2 REM**EL MISMO PROGRAMA QUE
  PARA EL C-64
3 REM**CAMBIANDO LAS
  SIGUIENTES LINEAS
10 POKE 36879,26:PRINT
```



```
    "[SHIFT+CLR/HOME]"
30 DIM S(308,2)
40 POKE 36878,15
135 POKE 36867,32:GOSUB 1500
140 T=1:H=3
145 YV=1:XV=0
150 FOR N=1 TO 3:SP=7680+S
    (N,1)*22+S(N,2):POKESP,
    163:POKE 30720+SP,6:NEXT
190 SP=PEEK(7680+Y*22+X):IF
    SP=160 OR SP=163 OR Y=0
    THEN 2000
200 H=H+1:IF H=308 THEN H=1
210 SP=7680+Y*22+X:POKE SP,
    163:POKE SP+30720,6
230 SP=S(T,1)*22+S(T,2)+7680
    :POKE SP,32
240 T=T+1:IF T=308 THEN T=1
245 IF P<>0 THEN 260
250 P=INT(RND(1)*9)+1:FY=INT
    (RND(1)*13)+2:FX=INT(RND
    (1)*18)+1
255 IF PEEK(7680+FY*22+FX)<>
    32 THEN 250
260 SP=7680+FY*22+FX:POKE SP
    ,P+48:POKE SP+30720,0
280 P=P-1:IF P=0 THEN POKE
    7680+FX+FY*22,32
300 S=S+P:SP=7680+Y*22+X:
    POKE SP,163:POKE SP+
    30720,6:PRINT
    "[CLR/HOME][CTRL+9]"S
325 SP=PEEK(7680+Y*22+X):IF
    SP=160 OR SP=163 OR Y=0
    THEN 2000
330 H=H+1:IF H=308 THEN H=1
350 SP=7680+S(H,1)*22+S(H,2)
    :POKE SP,163:POKE SP+
    30720,6
1500 FORN=0T021:POKE7680+N,
     160:POKE38400+N,0:POKE
     8010+N,160:POKE38730+N
     ,0
1510 POKE7680+N*22,160:POKE
     7701+N*22,160
1515 POKE 38400+N*22,0:POKE
     38421+N*22,0:NEXT
1520 PRINT"[CLR/HOME]
     [CTRL+9] [CTRL+1]"S;
     TAB(10)HS
2000 FOR Z=200T0127STEP-1:
     POKE 36877,Z:NEXT
2030 PRINT"[CLR/HOME][2*CRSR
     abajo]";TAB(9)"[CTRL+3]
     [CTRL+9]FUERA!"
2040 FOR F=0 TO 10:PRINT
     "[CLR/HOME][2*CRSR
     abajo]"TAB(9)"FUERA!":
     FOR G=0 TO 90:NEXT:
     PRINT"[CLR/HOME][2*CRSR
     abajo]"TAB(9)"[CTRL+9]
     FUERA!"
2050 FOR G=0 TO 90:NEXT G,F
2070 POKE 36867,174:PRINT
     "[SHIFT+CLR/HOME]
     [8*CRSR abajo][5*CRSR
     arriba]PUNTOS"S:PRINT
     "[2*CRSR abajo]MAYOR
     PUNTUACION"HS
2080 PRINT "[CLR/HOME]
     [CTRL+9]PULSA UNA
     TECLA PARA JUGAR DE
     NUEVO."
2090 POKE 198,0:WAIT 198,1:
     PRINT"[SHIFT+CLR/HOME]":
     GOTO 100
```



En la línea 10 se definen los colores de la pantalla y se borra ésta, dejándola lista para los gráficos del juego. La puntuación máxima se pone a cero en la línea 20, mientras que la puntuación actual se pone a cero en la línea 100.

La línea 30 dimensiona la matriz S, que se utilizará para almacenar las coordenadas de la serpiente sobre la pantalla. La máxima longitud de la serpiente es de 1000 segmentos para el **Commodore 64** y de 308 para el **Vic 20**; por ello, para definir cada posición de pantalla hace falta una matriz bidimensional de 1000×2 en el caso del **64** y de 308×2 elementos para el **Vic 20**, ya que para cada posición de pantalla se necesitan las coordenadas x e y. Inicialmente la serpiente ocupará las posiciones (10,14), (11,14) y (12, 14). Las líneas 110 a 130 sitúan las coordenadas en los primeros tres pares de elementos de la matriz.

La subrutina de definición de pantalla empieza en la línea 1500 y es llamada desde la línea 135. La línea 1500 dibuja los bordes superior e inferior de la pantalla y las líneas 1510 y 1515 dibuja los bordes laterales. La puntuación actual y la máxima se establecen en la línea 1520.

Las variables H y T que figuran en la línea 140 son los punteros de la matriz. H está apuntando al par de elementos que contienen las coordenadas de la cabeza mientras que T hace lo mismo para las coordenadas de la cola. En el caso de la línea 140, la cola está almacenada en el primer par y la cola se almacena en el tercer par.

En la línea 145 se definen dos vectores, xv e yv, encargados de llevar el seguimiento de la dirección en que se mueve la serpiente. Cualquiera de ellos puede tener uno de tres valores: El 0 significa que la serpiente no está mirando en esa dirección; el 1 significa que la serpiente sigue recta o va hacia abajo, y el −1 significa que sigue recta o va hacia arriba. La línea 145 hace entonces que se mueva la serpiente hacia arriba al principio del juego.

La línea 150 POKEa la serpiente en la pantalla. Hay un bucle FOR ... NEXT que toma los valores uno a tres, lo cual hace que la serpiente esté inicialmente formada por tres segmentos. La línea 160 utiliza dos variables: x e y, para hacer el seguimiento de la cabeza de la serpiente. A estas variables se les asigna la posición POKEada para la cabeza en la línea 150. La línea 165 contiene un indicador P, para saber cuándo está siendo presentado algún número.

La línea 175 llama a la subrutina de lectura del teclado, que empieza en la línea 1000. Las líneas 1010 a 1040 leen las teclas [Q], [A], [O] y [P] y establecen los vectores de forma apropiada. La línea 1050 utiliza estos vectores para calcular la posición de la serpiente y finaliza la subrutina.

La línea 190 comprueba si la serpiente ha invadido alguno de los bordes o se ha entrecruzado consigo misma. Caso de que ocurra alguna de estas dos cosas, salta a la subrutina de «Final de Juego» que está en la línea 2000. Si la posición de la serpiente es legal, la línea 200 incrementa el puntero de la cabeza y lo desplaza hacia atrás hacia el principio de la matriz en caso de que haya llegado al final. La cabeza es POKEada en su nueva posición por medio de la línea 210 y la línea 220 se encarga de introducir la nueva posición en la matriz. La línea 230 borra la cola, de forma que la serpiente no se vaya haciendo cada vez más larga. La línea 240 incrementa el puntero de la cola y comprueba que todavía está apuntando a un elemento situado dentro de la matriz.

Cuando está siendo presentado algún número, la línea 245 envía el programa a la 260. En caso contrario, habrá que presentar un nuevo número. La línea 250 selecciona un nuevo número aleatorio y una posición en que presentarlo. La línea 255 comprueba si la posición elegida no es un espacio prohibido. Si así es, el programa retrocede de nuevo a la línea 250. La línea 260 POKEa el número en la pantalla.

A medida que va transcurriendo el tiempo, el número que hay en la pantalla va siendo decrementado. En la línea 270 se introduce un cierto factor de aleatoriedad, comparando el número aleatorio con 0.95. En la mayoría de los casos, la línea de decremento de la puntuación, que es la 280, se salta. Lo que hace esta línea es comprobar que dicho número sigue siendo mayor que cero después de haber sido decrementado; en caso contrario dicho número se borra. Si la cabeza no ocupa la misma posición de pantalla que el número en cuestión, la línea 290 completa el bucle, enviando el programa nuevamente a la línea 170.

Cuando la serpiente se come el número, el programa llega a la línea 300. La puntuación se incrementa justamente en el valor del número que se acaba de comer. La cabeza se POKEa en la pantalla y aparece el valor del tanteo alcanzado.

El bucle FOR ... NEXT que se extiende entre las líneas 310 y 360 añade a la serpiente el número correcto de segmentos; este número de segmentos está determinado por el número que acaba de ser engullido por sus colmillos. Cada vez que se recorre el bucle, la línea 320 llama a la rutina de lectura del teclado; la línea 325 comprueba si la cabeza sigue estando todavía en una posición legal y la 330 incrementa el puntero de la cabeza. A continuación la línea 340 almacena en la matriz la nueva posición de la cabeza y POKEa dicha cabeza por medio de la línea 350. Para añadir los nuevos segmentos lo que se hace es no suprimir lo que sobra de la cola. Como el programa se encuentra ahora con menos cosas que hacer, el movimiento de la serpiente se verá considerablemente acelerado, pero introduciendo el retardo de la línea 355, la velocidad se mantendrá constante.

La sección final del programa, que empieza en la línea 2000, es la rutina de fin de juego. En el caso del **Commodore 64**, la línea 2000 deja a la máquina lista para generar un efecto sonoro. Las líneas 2010 a 2020 son las que generan dicho efecto. En el **Vic 20**, el efecto sonoro se genera únicamente en la línea 2000. No te olvides de subir el volumen del televisor. Las líneas 2030 y 2040 hacen que aparezca intermitentemente el mensaje FUERA! en el borde superior. Hay una pequeña pausa en la línea 2050 antes de que el sonido se extinga. La línea 1060 actualiza la puntuación máxima en caso de que sea necesario y la 2070 presenta la puntuación actual y la máxima.



ASI ES EL MAPA DE MEMORIA DEL COMMODORE 64 (III)

Bytes $0039 y $003A = 57 y 58

Aquí se guarda el número de línea con la que está trabajando el programa en BASIC en el momento presente.

A medida que el programa avanza en su ejecución este número va siendo actualizado automáticamente.

Bytes $003B y $003C = 59 y 60

Estos dos bytes almacenan el número de línea de programa procesada previamente.

Bytes $003D y $003E = 61 y 62

Este puntero es de algún modo complemento a las direcciones $39 y $3A. Almacena la dirección de la línea de texto en BASIC que se está procesando. No confundirlo con el número de línea, puesto que se trata de la dirección de la memoria del ordenador en la cual comienza la línea.

Bytes $003F y $0040 = 63 y 64

Cuando existen líneas con DATA en un programa se almacena en estas direcciones el número de línea al que se está dirigiendo en cada momento el programa para la lectura de datos.

Bytes $0041 y $0042 = 65 y 66

Es un puntero que indica la dirección del siguiente elemento DATA que debe ser leído.

Bytes $0043 y $0044 = 67 y 68

Este puntero indica la dirección de memoria del origen de la fuente de datos que utilizan en cada momento INPUT, READ o GET.

Bytes $0045 y $0046 = 69 y 70

Aquí se guarda el nombre de la variable (2 bytes) que se está buscando en el momento en que es necesaria (consultar bytes $2D y $2E).

Bytes $0047 y $0048 = 71 y 72

Apuntador hacia el byte anterior al descriptor de la variable utilizada en cada instante por el programa en BASIC.

Bytes $0049 y $004A = 73 y 74

Contiene la dirección de una variable utilizada en un bucle FOR...NEXT en ese instante.

Bytes $004B y $004C = 75 y 76

Durante las operaciones matemáticas de comparación se utilizan estas direcciones como memoria intermedia.

Byte $004D = 77

Cuando se realiza una operación de comparación se utiliza esta dirección para almacenar una máscara que determina si el resultado es mayor que, menor que o igual.

Bytes $004E y $004F = 78 y 79

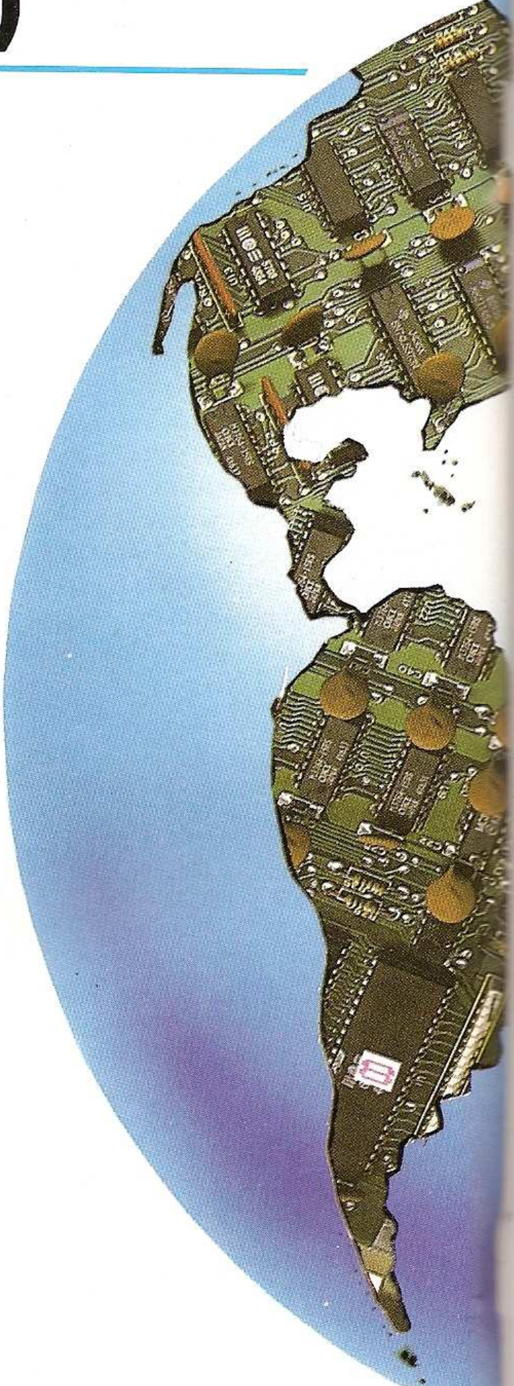
Estas direcciones están relacionadas con la definición de funciones —DEF FN— actuando como puntero hacia el descriptor correspondiente.

Bytes $0050 a $0053 = 80 a 83

Puntero hacia el descriptor de cadenas utilizado en ese instante. También almacena la longitud de la cadena.

Byte $0054 = 84

Contiene el valor constante $4C, que es el primer byte de la instrucción de salto (JMP) utilizada en el microprocesador 6510 para saltar incondicionalmente a cualquier dirección de la memoria. Completa el salto con una dirección procedente de la tabla que comienza en $A052.



Bytes $0055 y $0056 = 85 y 86



Vector del salto descrito en el anterior byte.

Bytes $0057 a $005B = 87 a 91

Registro utilizado en opraciones aritméticas, actúa como acumulador número 3.

Bytes $005C a $0060 = 92 a 96

Registro utilizado en operaciones aritméticas, actúa como acumulador número 4.

Bytes $0061 a $0066 = 97 a 102

Acumulador número 1 para operaciones en coma flotante. Se utiliza también en conversiones de datos a distintos formatos (p.ej. coma flotante a enteros o viceversa).

Byte $0067 = 103

Actúa como contador en una evaluación de polinomios. Recoge el número de evaluaciónes individuales que deben efectuarse para resolver una expresión.

Byte $0068 = 104

Byte de redondeo para el acumulador número 1 en coma flotante.

Bytes $0069 a $006E = 105 a 110

Acumulador número 2 en coma flotante. Su formato (exponente, mantisa, signo) es el mismo para el acumulador número 1 ($61 a $66).

Byte $006F = 111

Contiene el resultado de la comparación de signos entre los contenidos del acumulador número 1 y el acumulador número 2.

Byte $0070 = 112

Byte de redondeo para el acumulador de coma flotante número 1.

Bytes $0071 y $0072 = 113 y 114

Puntero el *buffer* del *cassette*.

RELACIONES LOGICAS

- OPERADORES RELACIONALES
- ¿UNA COSA ES MAYOR, IGUAL O MENOR QUE OTRA?
- OPERADORES LOGICOS
- USO DE AND, OR Y NOT

Los ordenadores son capaces de ejecutar millones de operaciones lógicas por segundo. Pero la lógica es además una parte fundamental de cualquier programa.

Hay tres tipos de expresiones que se utilizan en la programación con BASIC. Una gran parte de todo programa está constituída por operaciones aritméticas y las realizadas con cadenas de caracteres; sin embargo, es el tercer tipo de operaciones, las realizadas con las expresiones lógicas, el que realmente se ocupa de la toma de decisiones dentro de un programa.

Su función elemental consiste en evaluar si algo es «verdadero» o «falso». Las palabras y símbolos que utiliza el programa para hacer esto se llaman operadores (en algunos casos se llaman también conectores.

En las expresiones lógicas se utilizan dos tipos de operadores: los relacionales (que incluyen símbolos matemáticos tales como <, > e =), y los lógicos que forman parte del repertorio de palabras reservadas en todas las versiones del BASIC: AND, OR y NOT.

MAYOR, IGUAL O MENOR

Los operadores relacionales pueden utilizarse incluso en los programas BASIC más sencillos. Las posibles comparaciones que utilizan estos operadores son:

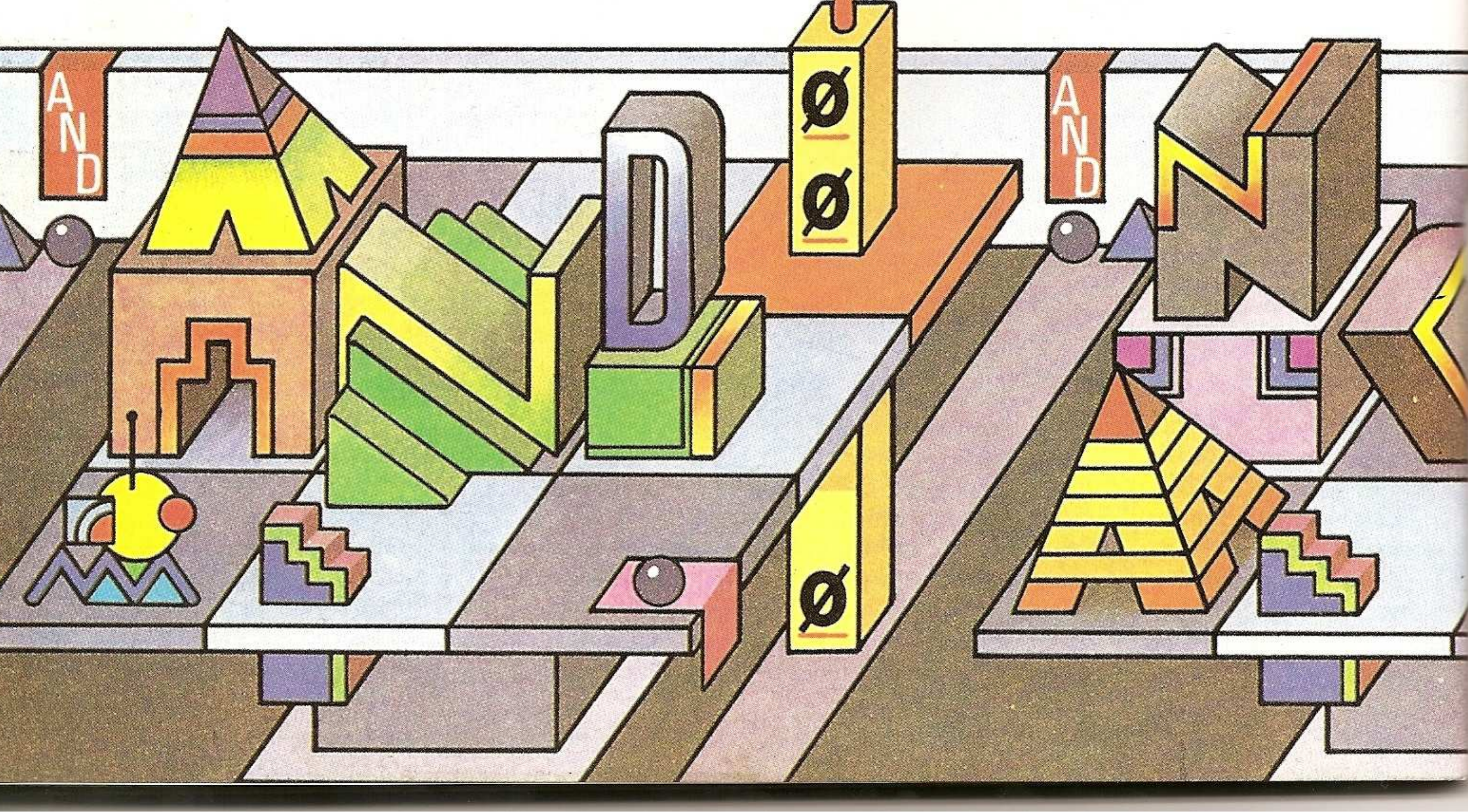
A > B ... A es mayor que B
A < B ... A es menor que B
A ≥ B ... A es mayor o igual que B
A ≤ B ... A es menor o igual que B
A = B ... A es igual que B
A <> B ... A es diferente de B

Habrás visto estos operadores utilizados docenas de veces en los programas de INPUT o en cualquier otra parte, y aunque su utilización es múltiple, su valor en las pruebas condicionales resulta más evidente en relación con la sentencia IF ... THEN. Un ejemplo típico de esto es:

```
IF A>B THEN PRINT "A ES
MAYOR QUE B"
```

En este caso, el valor de la variable A debe superar el valor de B para que se ejecute el resto de la línea. En los casos en que el valor de A permanezca por debajo del valor de B, el programa se limitará a pasar a la línea siguiente. Puedes ver que es posible disponer de una cierta capacidad de salto condicional: si resulta un determinado conjunto de valores, el programa hace una cosa; si resulta otro conjunto de valores, el programa hace otra cosa distinta.

Los operadores relacionales no están únicamente limitados a manejar valores numéricos; también sirven para comparar cadenas de caracteres. No obstante, hay que manejarlos con precaución para evitar encontrarte con resultados inesperados. Se debe tener presente que la comparación siempre se detiene en el número de caracteres correspondiente a la más corta de las dos cadenas que se compa-



ran. Así, si una de las cadenas contiene once caracteres, mientras que la otra sólo siete, únicamente intervendrán en la comparación los primeros siete caracteres de la cadena más larga, comparándose con sus correspondientes en la cadena corta.

Los caracteres se comparan uno por uno y de izquierda a derecha; de aquí pueden surgir los resultados inesperados. En una comparación de tipo numérico, la proposición 5 > 10 es evidentemente falsa, pero en una comparación entre cadenas, puedes encontrarte una sentencia en la que se comparen dos variables en la siguiente forma: A$ > B$. Si A$ = «5» y B$ = «10» el ordenador compara en primer lugar el carácter que figura más a la izquierda, que en este caso es un 5 en una de las cadenas y un 1 en la otra, y a continuación se para porque considera que ya no tiene nada que comparar.

En consecuencia se obtiene un «verdadero» como resultado de la comparación, debido a que 5 es mayor que 1, pero el ordenador ignora los números restantes de la cadena más larga. Tienes que estar atento para no cometer errores de este tipo.

En las comparaciones entre cadenas de caracteres, a las letras del alfabeto se les supone la siguiente relación de orden: «A» < «B» < «C» < «D» , etcétera. Pero también aquí has de tener cuidado, ya que el ordenador no entiende realmente lo que estos caracteres significan. Lo que hace es comparar los códigos correspondientes a cada carácter, siendo igualmente significativos dentro de la cadena los espacios y otros caracteres que no son letras, pero que disponen también de un código propio. Como puedes imaginarte, esto podría causar estragos en un programa de clasificación de cadenas por orden alfabético.

Cuando ambas cadenas contienen una secuencia de caracteres parcialmente idéntica, se considera numéricamente mayor a la cadena más larga. Pero observa que una cadena más corta es considerada mayor en una expresión como la siguiente: «ABD» > «ABCD» ya que la comparación se detiene en cuanto se encuentra con la primera diferencia, es decir, al comparar los códigos correspondientes a los caracteres «D» y «C». El funcionamiento es de hecho el mismo que se sigue al asignar prioridades alfabéticas en la redacción de un índice o un diccionario, en el que «para» figura antes que «paradoja», pero después de «pan».

VERDADERO O FALSO

Toda comparación produce un resultado que en realidad es un número entero. Cuando la comparación resulta ser falsa, el resultado es 0, mientras que cuando es verdadera el resultado es 1, dependiendo del ordenador que sea. Teclea este comando en modo directo para observar el resultado que corresponde a tu ordenador:

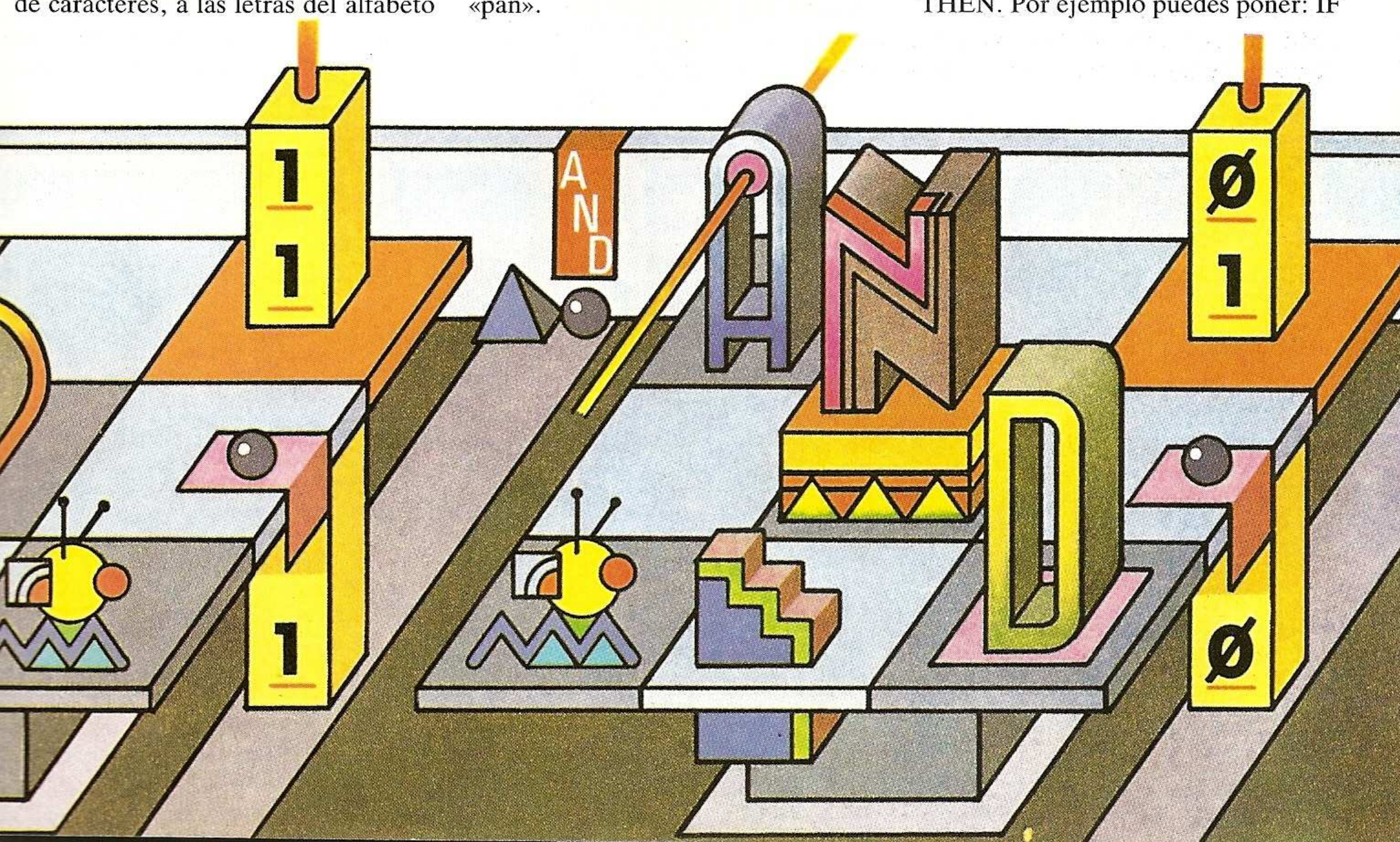
```
PRINT 6>5, 5>7
```

La expresión de la izquierda es verdadera y la de la derecha falsa. En consecuencia obtendrás en la pantalla un 1 o un 0.

Los números enteros que obtengas como resultado de las comparaciones, puedes utilizarlos para realizar cálculos dentro de tus programas. No obstante tienes que tener cuidado de no intentar hacer divisiones por 0, lo que te daría un mensaje de error.

OPERADORES LOGICOS

Las **palabras reservadas** AND, OR y NOT son operadores lógicos que constituyen una poderosa extensión a la capacidad de toma de decisiones por medio de la sentencia IF ... THEN. Por ejemplo puedes poner: IF



esto es cierto AND eso otro es cierto, THEN hacer una determinada cosa. De esta forma puedes ir construyendo funciones de gran complejidad. Estos operadores, que a veces se llaman también operadores **booleanos** (aunque no debes dejarte impresionar por este nombre) se pueden utilizar en comandos directos o dentro de los programas para obtener un «valor de verdad» en condiciones de prueba que pueden llegar a ser muy complejas. De esta forma constituyen una alternativa muy cómoda a lo que en otro caso sería un confuso conglomerado de saltos condicionales a base de una utilización masiva de sentencias IF ... THEN.

USO DE «AND»

En su forma más simple, se puede considerar que el operador AND tiene el mismo significado que la conjunción española «y». En una expresión tal como la siguiente:

```
IF V>0 AND V<100 PRINT
"DENTRO DE RANGO"
```

El mensaje sólo aparecerá en pantalla cuando la variable V esté comprendida entre 0 y 100.

En un programa puedes tener una línea como ésta:

```
990 OKAY=1 AND MES>5 AND
MES<10 AND ANO>1900 AND
ANO<2001
```

Esta línea de programa te permitiría comprobar si una determinada fecha corresponde a un verano del siglo XX.

En este caso se utiliza AND para realizar cuatro pruebas, cuyos resultados deben ser todos ciertos para que OK siga siendo verdadera. En esta prueba de validación se asigna en primer lugar a la variable OK el valor «verdadero». A continuación se impone la condición de que MES caiga en el intervalo de 6 a 9, y de que AÑO esté entre 1901 y 2000. Pero miremos un poco más en detalle lo que ocurre. Si se cumplen todas las condiciones, todas las expresiones darán un valor verdadero. Así, de hecho la línea de programa se convierte en :

```
990 OKAY=-1 AND -1 AND -1
AND -1 AND -1
```

Si ahora le añades PRINT OK, puedes comprobar que el resultado es en efecto 1, es decir verdadero.

USO DE «OR»

El operador OR puede recibir una consideración en gran parte análoga a la de AND, aunque su significado es diferente, como ocurre con la conjunción española «o» en su uso cotidiano. También aquí observaremos su empleo en una sencilla línea de programa en la que se utiliza para comparar los valores de una variable:

```
IF V=8 OR V=10 PRINT "OKAY"
```

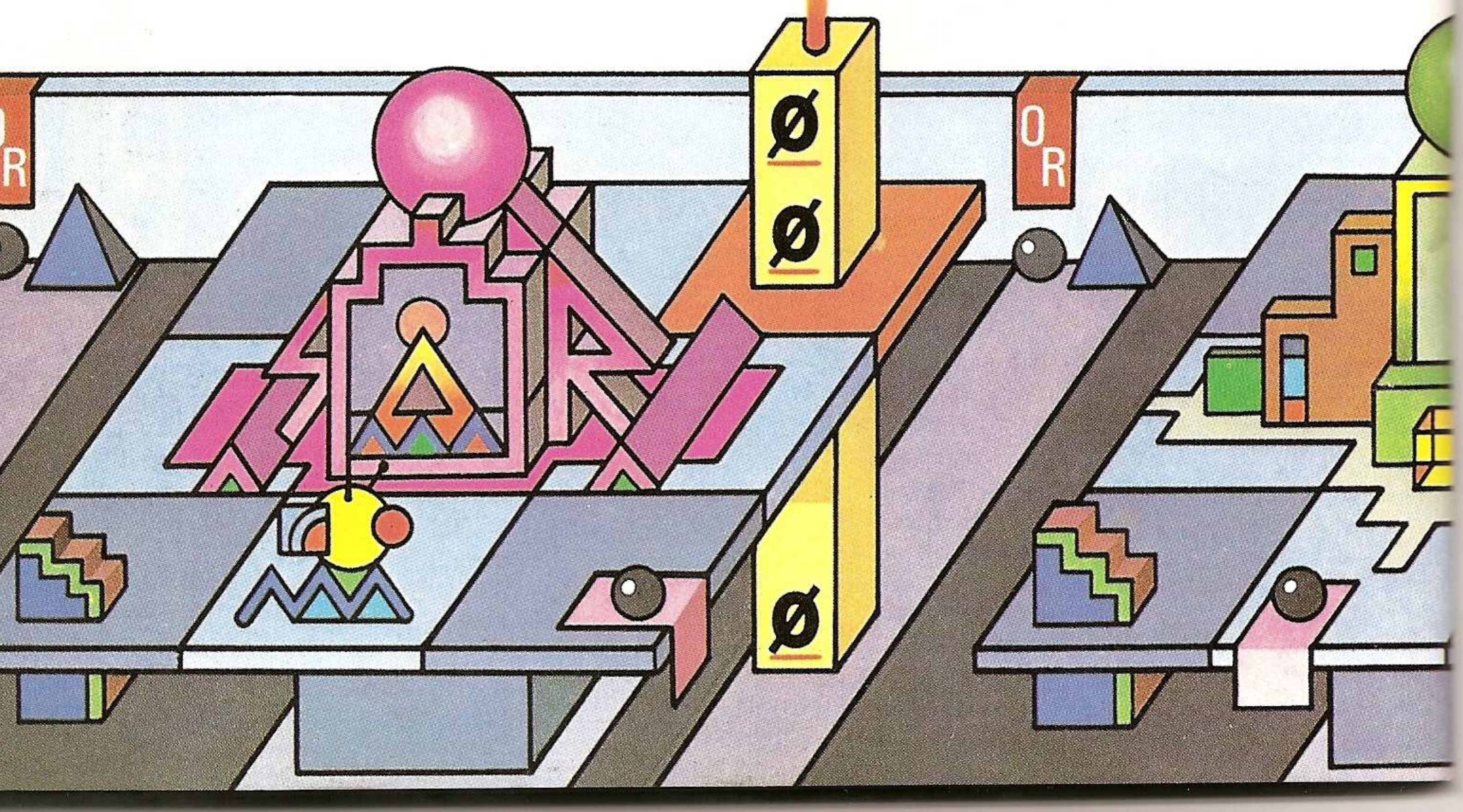
El mensaje aparecerá en la pantalla si el valor de la variable V es 8 o 10.

El operador OR resulta de gran utilidad cuando se comprueba la validez de las entradas suministradas a una sentencia INPUT dentro de un programa. Si por ejemplo el programa te pide que introduzcas tu edad, siempre habrá alguien que tecleará —10 ó 999 con la intención de probar a ver qué pasa. Pero puedes solucionar el problema de la siguiente forma:

```
10 INPUT A
20 IF A<1 OR A>120 THEN
PRINT "ESTAS DE BROMA?":
GOTO 10
```

USO DE «NOT»

El tercer operador lógico que se utiliza con frecuencia es el NOT. Es un poco diferente de los anteriores en el sentido de que sólo utiliza un argumento, es decir, sólo actúa sobre la expresión numérica o lógica que le sigue,



a diferencia de AND y OR que actúan sobre la expresión que les precede y sobre la que les sigue, es decir son operadores binarios o de dos argumentos.

Puedes utilizar NOT en un programa como el siguiente:

```
IF NOT (A>10) THEN 999
```

En el lenguaje hablado normal, podríamos traducir la anterior expresión más o menos así: «Si el valor de A no es mayor que 10, pasa a la línea 999».

La función lógica que realiza el operador NOT es convertir una condición verdadera en falsa y viceversa. Puedes utilizarla pues para invertir el valor resultante de una prueba anterior.

TABLAS DE VERDAD

En relación con los operadores lógicos, te encontrarás con mucha frecuencia con las «tablas de verdad».

En realidad son algo muy sencillo que se utiliza para dar una representación visual de lo que ocurre cuando se hacen operaciones de AND y OR con los valores 1 y 0 (verdadero y falso). El operador NOT no necesita en realidad una tabla de verdad, ya que basta con invertir los valores para observar su efecto.

Las tablas de verdad pueden adoptar formas diferentes. Una forma típica para el operador AND es:

A	B	C	
-1	-1	-1	(linea 1)
-1	0	0	(linea 2)
0	-1	0	(linea 3)
0	0	0	(linea 4)

El significado de la línea 1 de la tabla es el siguiente: «Si A es verdadera y B es verdadera, entonces C es verdadera». El significado de la línea 2 es: «Si A es verdadera y B es falsa, entonces C es falsa». La línea 3 significa: «Si A es falsa y B es verdadera, entonces C es falsa». La línea 4 por su parte significa: «Si A y B son falsas, entonces C es falsa». Análogamente la tabla de verdad del operador OR es:

A	B	C
-1	-1	-1
-1	0	-1
0	-1	-1
0	0	0

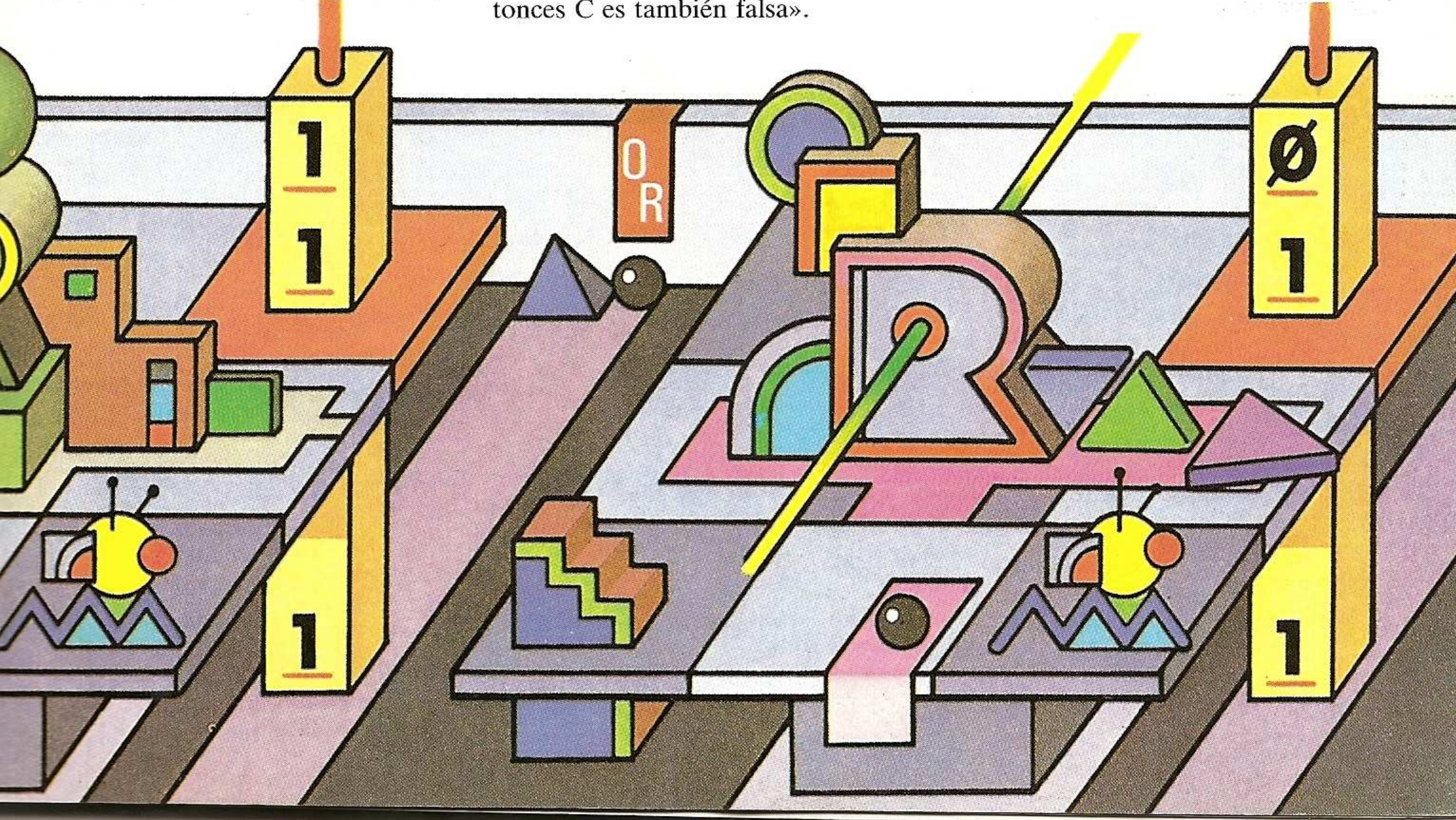
La primera línea significa: «Si A es verdadera y B es verdadera, entonces C es verdadera». Las líneas 2 y 3 pueden ser ambas interpretadas como: «Si A o B son verdaderas, entonces C es verdadera». El significado de la línea 4 es: «Si A y B son ambas falsas, entonces C es también falsa».

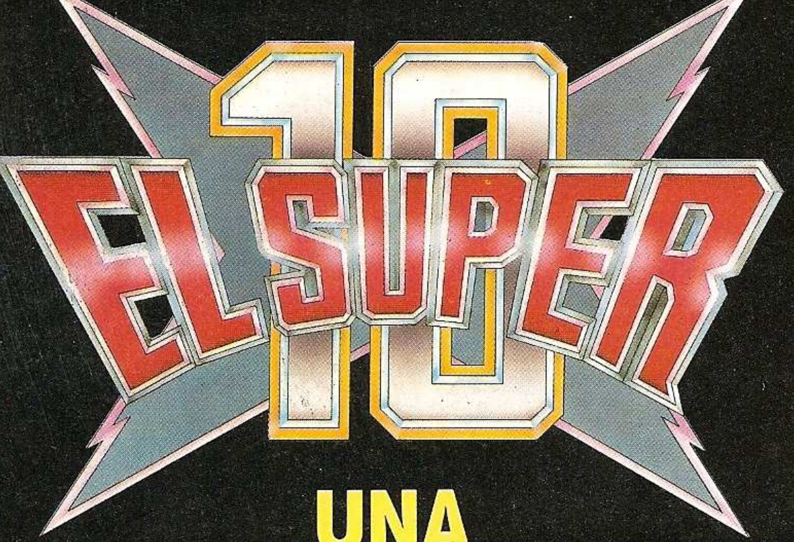
APLICACION PRACTICA

Aquí tienes finalmente un programa que utiliza los operadores AND, OR y NOT. Se trata de una versión simplificada de un programa que calcula la escala correcta de salarios para alguien que ha solicitado un empleo:

```
10 INPUT "EDAD";EDAD
20 INPUT "NUMERO DE NIVELES"
   ;NIVELES
30 LET MAS18=(EDAD>=18)
40 LET CUAL=(NIVELES>=5)
50 IF NOT MAS 18 AND NOT
   CUAL THEN PRINT "NO
   UTILIZABLE"
60 IF (NOT MAS18 AND CUAL)
   OR (MAS18 AND NOT CUAL)
   THEN PRINT "ESCALA
   SALARIAL UNO"
70 IF MAS18 AND CUAL THEN
   PRINT "ESCALA SALARIAL
   DOS"
```

Supongamos que como respuesta a las dos primeras sentencias INPUT, tecleas 20 años de edad y cuatro niveles O. Esto significa que (EDAD ⩾ 18) es verdad, mientras que (ONIVEL ⩾ 5) es falso. En consecuencia la persona es MAS18 (más de 18 años) y NO CUAL (no cualificada). El ordenador





UNA OPORTUNIDAD IRREPETIBLE

Ahora tienes la ocasión de hacerte con los 10 mayores éxitos del año en su presentación original (cada uno en su estuche y con su carátula) a un precio de auténtico chollo: 3.995 pts.

Imagínate... Los 10 mejores títulos de 1986 por poco más de lo que cuesta uno solo.

¡¡¡PIDE EL SUPER-10 EN TU TIENDA ANTES QUE SE AGOTE!!!

"SUPER-10" SPECTRUM	"SUPER-10" AMSTRAD
EXPLODING FIST	EXPLODING FIST
TURBO ESPRIT	TURBO ESPRIT
ROCK'N LUCHA	ROCK'N LUCHA
ZORRO	ZORRO
3 WEEKS IN PARADISE	3 WEEKS IN PARADISE
ABU SIMBEL (PROFANATION)	ABU SIMBEL (PROFANATION)
SABOTEUR	SABOTEUR
CAULDRON II	CAULDRON II
BRUCE LEE	BRUCE LEE
SPY HUNTER	RAID OVER MOSCOW

"SUPER-10" COMMODORE	"SUPER-10" M.S.X.
EXPLODING FIST	ALIEN-8
URIDIUM	KNIGHT LORE
GOONIES	GUNFRIGHT
SABOTEUR	NIGHTSHADE
BEACH HEAD II	JACK THE NIPPER
CRITICAL MASS	SHOWJUMPER
SPY HUNTER	VALKYR
ZORRO	BOUNDER
SUPER-ZAXXON	MAPGAME
FIGHTING WARRIOR	JET SET WILLY II

DISPONIBLE EN
SPECTRUM • COMMODORE
AMSTRAD • MSX

LOS 10 EXITOS DEL AÑO

encuentra enseguida este conjunto de condiciones en la línea 60, imprimiendo a continuación «ESCALA DE SALARIO UNO». Puedes ampliar fácilmente este programa para que compruebe más condiciones, por ejemplo si el solicitante cuenta con más de dos años de experiencia, o cualquier otra circunstancia que sea necesaria para el trabajo.

OPERACIONES NUMERICAS

El uso de los operadores lógicos no está únicamente limitado a las sentencias IF ... THEN que acabamos de ver. También se pueden utilizar en algunos tipos de operaciones numéricas. Puede que ya las hayas visto utilizadas así en algún programa y a lo mejor te has quedado maravillado de lo que sucedía. De hecho no siempre funcionan de la forma obvia que podría esperarse. Teclea lo siguiente:

```
PRINT 375 AND 47
```

Tal vez esperas que el resultado sea 422, o quizás 37547. Y sin embargo el resultado es 39. ¿Qué ha sucedido? Para explicarlo, tenemos que adentrarnos algo más en la forma de trabajar del ordenador, a fin de entender el modo en que éste maneja los dos argumentos cuyo AND se calcula.

En principio ambas expresiones (375 y 47) se convierten en su equivalente binario de dos bytes.

375 en binario es:
0000000101110111

Por su parte, 47 en binario es:
0000000000101111

A continuación la función AND realiza una comparación bit a bit sobre los 16 bits, dando un 1 como resultado únicamente cuando hay un 1 en el mismo bit de ambos números en binario. Empezando por la derecha, sólo los pares de bits cero, 1, 2 y 5 cumplen esta condición. Esto da como resultado el número binario 0000000000100111.

Al volver a hacer la conversión a decimal, los bits 5, 2, 1 y 0 puestos a 1 dan un valor combinado de 39; éste es el resultado de hacer un AND de 375 y 47.

Ensaya ahora el siguiente comando:

```
PRINT 375 OR 47
```

¿A qué se debe el resultado 383 que se obtiene en este caso? Al realizar la operación con OR lógico el resultado de cada comparación bit a bit es 1 si hay un 1 en cualquiera de los bits del par que se compara.Observa de nuevo la forma binaria de los números anteriores y verás que los bits 8, 6, 5, 4, 3, 2, 1 y 0 tienen un 1 en al menos uno de los bits del par. El número binario resultante es 0000000101111111 que es 383 en decimal.

Con el operador OR es posible obtener el mismo valor de bits con expresiones diferentes. Por ejemplo, si tecleas PRINT 315 OR 75, el resultado es también 383. Si quieres puedes comprobarlo realizando la conversión de los números a forma binaria.

El valor −1 (que se almacena como FFFFFFFF en hexadecimal, es decir un conjunto de bits que son todos 1) queda inalterado cuando con él se realiza una operación de OR lógico con otro número cualquiera. Para hacer la prueba, teclea el siguiente comando directo:

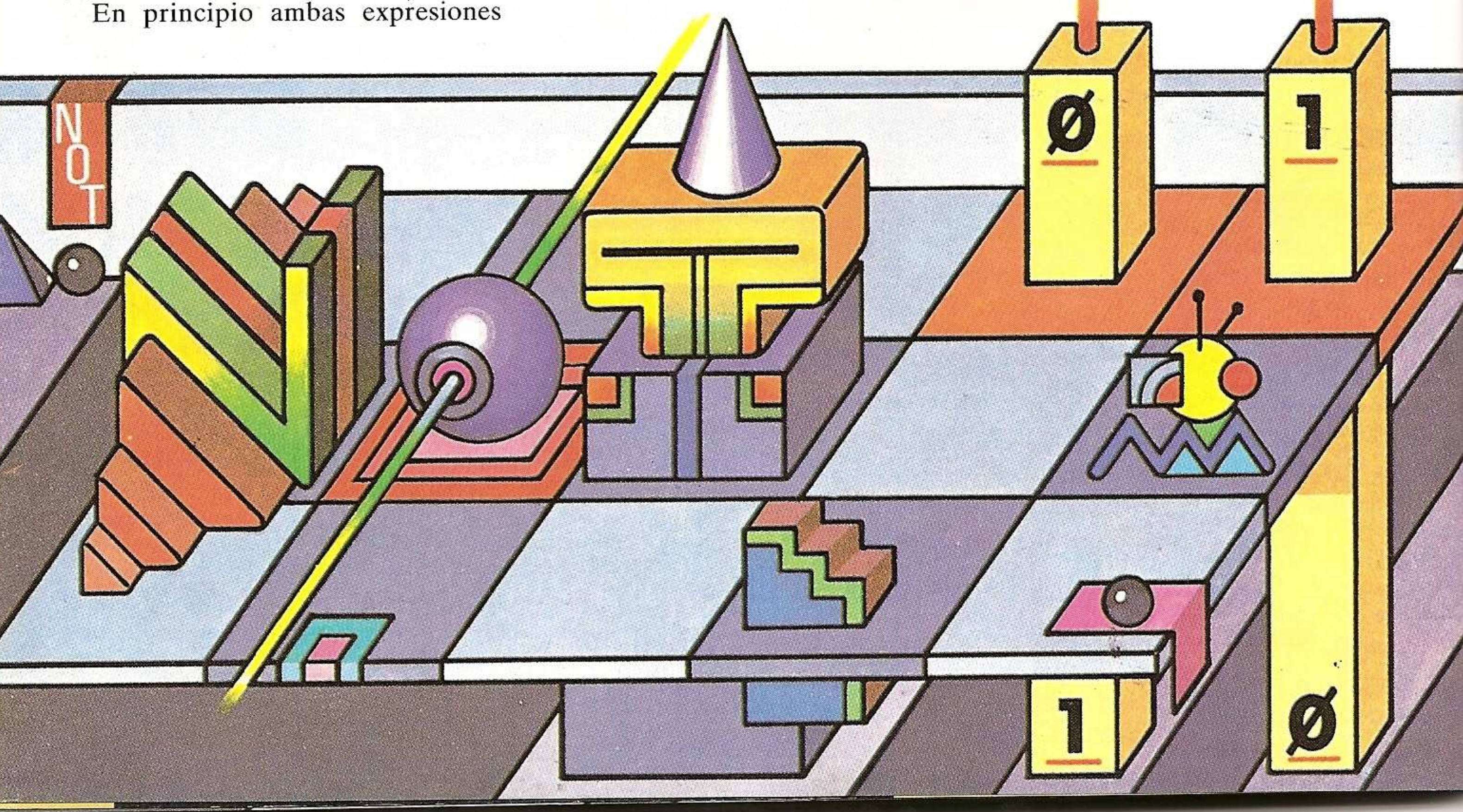
```
PRINT -1 OR 375
```

cuyo resultado es −1.

Veamos ahora una aplicación numérica para el operador NOT:

```
PRINT NOT 10.75,NOT -11
```

Los valores resultantes son −11 y 10. Así vemos que en efecto NOT suma 1 al número sobre el que se aplica y a continuación cambia su signo. Observa que el número en punto flotante se redondea por defecto al entero má próximo y que es posible estimar el valor real por medio de X = −(X + 1). De hecho este valor se convierte a un entero de cuatro bytes, todos cuyos bits quedan invertidos.



NUMEROS BAJO CERO

- CUANDO LOS NUMEROS NEGATIVOS SE HACEN NECESARIOS
- PROGRAMA DE CONVERSION PARA NUMEROS NEGATIVOS
- LA CONVENCION DEL SIGNO

La comprensión de los números binarios y hexadecimales no suele ser difícil, pero la expresión de valores negativos con estos sistemas de numeración puede presentar problemas.

En la programación de algunos juegos puede ser necesario utilizar valores negativos, por ejemplo, para desplazarse sobre la pantalla en determinadas direcciones. Las direcciones van codificadas en grupos de ocho bits, ya que esa es la única manera en que el ordenador puede almacenar los datos en la memoria. Pero surge un problema: como hemos visto, un byte puede representar cualquier número desde 0 hasta 255, es decir, desde 0000 0000 hasta 1111 1111. Pero con esto se agotan las posibilidades de un número binario de ocho bits, que ya no tiene sitio para un signo menos o más, ni forma alguna de representarlo.

En la aritmética ordinaria, al restar 1 de 0 se obtiene el valor −1; aquí tienes ese mismo cálculo realizado con números binarios de ocho bits:

```
0000 0000

      - 1
---------
1111 1111
```

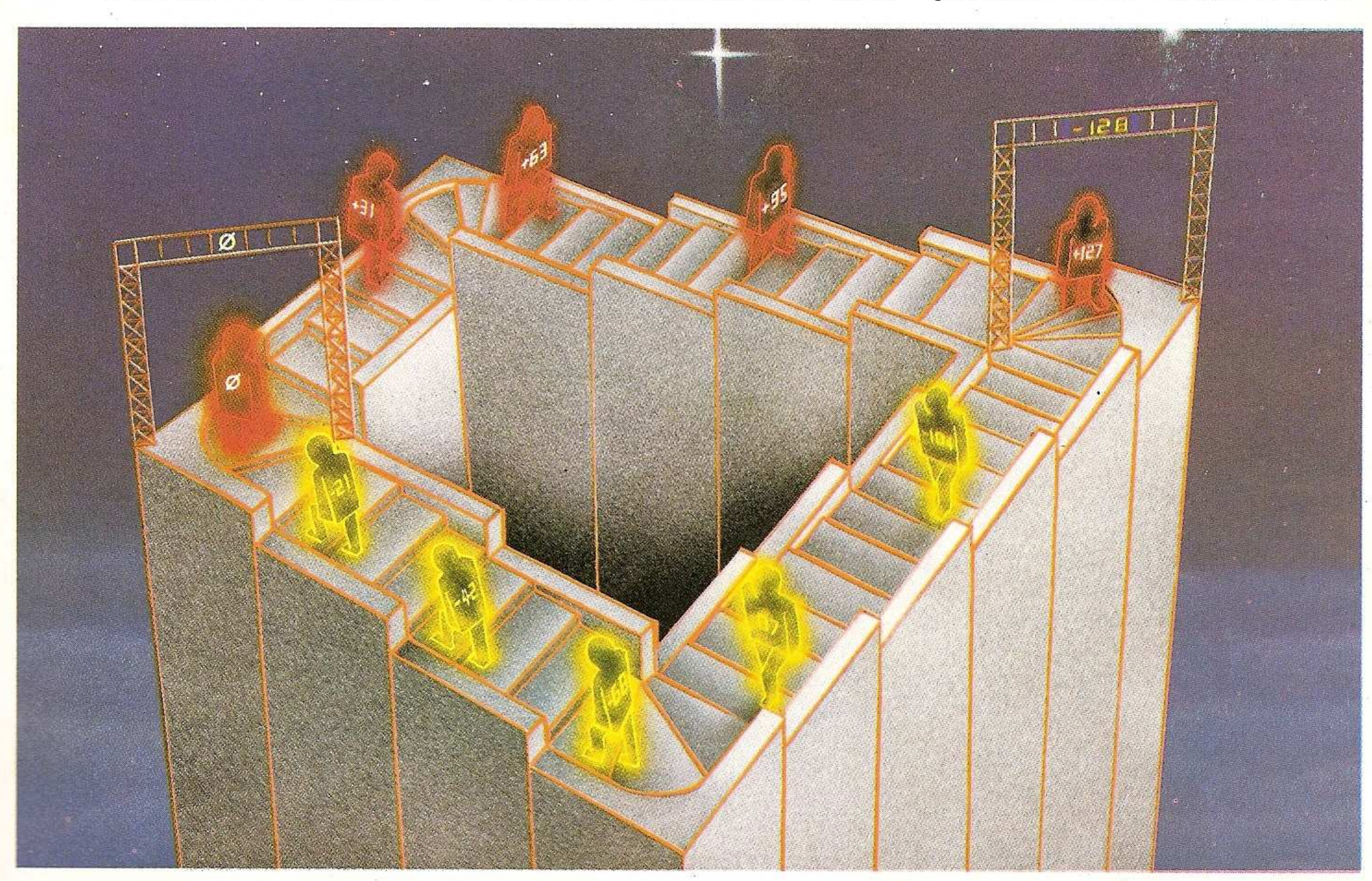
Puedes probar a tomar un 1 prestado de la columna situada más a la izquierda, pero cuando el número binario está limitado a ocho bits no hay manera de hacer nada. Análogamente, al restar otro 1 (en la aritmética tradicional se obtendría como resultado un −2) tenemos 1111 1110, que en binario corresponde al número decimal 254, ya que 1111 1111 es 255.

Esto no son curiosidades típicas de los números binarios: imagínate por ejemplo lo que ocurriría en decimal si sólo hubiese tres sitios o columnas en las que poner los números. Mira lo que sucedería si se intentase sumar 100 a 999 con estas limitaciones:

```
   100

+  999
------
(1)099
```

El uno que figura en la columna de las unidades de mil, está escrito entre paréntesis; en un sistema aritmético



con sólo tres columnas no hay espacio para las que «te llevas». En un sistema de sólo tres lugares, el resultado de esta suma es pues 99, o sea, exactamente el resultado de 100 menos 1.

De la misma forma, sumar 98 con 100 en un sistema de tres lugares equivale a restar 2, mientras que restar 998 de 100 equivale a sumarle 2.

En conclusión, una misma sucesión de números de un byte puede representar un número positivo o negativo, sin tener en cuenta la posible confusión o dificultad en el cálculo.

¿Qué hacer entonces? En la mayoría de las aplicaciones de los ordenadores domésticos, este problema no suele plantearse: tanto las direcciones de memoria como los códigos de operación, ambos expresados en binario, pueden considerarse siempre como positivos. Las únicas ocasiones en que se encuentran números negativos son en los datos o en los saltos del código máquina, equivalentes a los GOTOs del BASIC.

CAMBIANDO LOS BITS

Para pasar de un valor positivo en binario a su correspondiente negativo, hay que aplicar el llamado procedimiento del complemento a 2; no tiene una base teórica de fácil comprensión, pero sin embargo funciona bien.

Para pasar desde un número en binario a su correspondiente valor negativo, hay que cambiar los ceros por unos y viceversa, sumando al final 1 al resultado.

El siguiente programa realiza precisamente esta función. Observa que cuando el número en binario está limitado a ocho cifras, su equivalente en hexadecimal consta exactamente de dos cifras: esto quiere decir que también el equivalente hexadecimal del complemento a 2 es el correspondiente valor negativo.

TECLEA C-64

```
1 REM**[]SIGNIFICA ESPACIO
20 POKE 54277,33:POKE 54278,
   255:POKE 54273+23,15:POKE
   276,33:POKE 54273,0
30 FOR Z=1 TO 8:READ A(Z):
   NEXT Z:DATA 128,64,32,16,
   8,4,2,1
40 Z$="0123456789ABCDEF":A=0
   :A$="+-":AA=1:POKE 650,
   128:V$="[8*CRSR dcha.]"
50 POKE 53280,1:POKE 53281,1
   :PRINT"[SHIFT+CLR/HOME]
   [CTRL+1][4*CRSR abajo]"
60 FOR Z=1 TO 9:PRINT TAB(5)
   "[CTRL+9][29*ESPACIO]":
   NEXT Z
70 PRINT "[CTRL+3]
   [CRSR abajo]" TAB(9)
   "COMPLEMENTO A DOS"
80 PRINT "[CLR/HOME]" TAB
   (12)"NUMEROS NEGATIVOS"
90 PRINT TAB(12)"-----------
   ------"
100 PRINT "[CTRL+5][][]DEC
    [14*ESPACIO]BIN
    [14*ESPACIO]HEX"
110 PRINT"[FLECHA izq.]
    [4*CRSR abajo][8*RSR
    dcha.]COMPLEMENTO":PRINT
    "[3*CRSR abajo]"TAB(29)"
    +1"
120 PRINT"[4*CRSR arriba]
    [2*CRSR dcha.]+"TAB(37)
    "+"
130 PRINT"[9*CRSR abajo]----
    ------------------------
    ------------";
140 PRINT"[][]O[6*ESPACIO]
    O[][]O[][]O[][]O[][]O[][]
    O[][]O[][]O[6*ESPACIO]
    OO":GOTO 300
150 GET K$:IF K$="" THEN 150
160 S=0:SS=0:IF K$<>" " AND
    K$<>"B" THEN 500
170 A1=A:IF K$<>" " THEN 210
180 IF AA=2 THEN A1=A1-1
190 IF AA=1 THEN A1=A1+1
200 GOTO 240
210 IF K$<>"B" THEN 240
220 IF AA=2 THEN A1=A1+1
230 IF AA=1 THEN A1=A1-1
240 IF A1<0 OR A1>128 THEN
    AA=AA+1
250 IF A1>128 THEN A1=127
260 IF A1<0 THEN A1=1
270 A=A1
280 IF A=0 THEN AA=1
290 IF A=128 THEN AA=2:POKE
    54273,9:FORZ=.1TO20:NEXT
    :POKE 54273,0
300 IF AA>2 THEN AA=1
310 W=0:IF AA=1 THEN T=1:
    TT=0
320 IF AA=2 THEN T=0:TT=1:
    W=1
330 A1=A-W:FOR Z=1 TO 8:IF
    A(Z)<=A1 THEN B(Z)=T:
    C(Z)=TT:A1=A1-A(Z):GOTO
    350
340 B(Z)=TT:C(Z)=T
350 IF B(Z)=1 THEN S=S+A(Z)
360 NEXT Z:A2=0:FORZ=1 TO 8:
    IF C(Z)=0 THEN A2=A2+A(Z)
370 NEXT Z:A1=(255-A2)+T:IF
    A1>255 THEN A1=0
380 FOR Z=1 TO 8:IF A(Z)<=A1
    +W THEN D(Z)=1:A1=A1-A
    (Z):SS=SS+A(Z):GOTO 400
390 D(Z)=0
400 NEXT Z
410 PRINT"[CLR/HOME]
    [4*CRSR abajo]";MID$(A$,
    AA,1)"[4*ESPACIO]
    [4*CRSR izq.]"A
420 PRINT"[CRSR arriba]"V$;:
```



```
    FOR Z=1 TO 8:PRINT B(Z);
    :NEXT Z
430 ZA=INT(S/16):ZB=S-(16*ZA
    ):PRINT"[5*RESPACIO]"
    MID$(Z$,ZA+1,1);MID$(Z$,
    ZB+1,1)
440 PRINT "[4*CRSR abajo]
    "V$;:FOR Z=1 TO 8:PRINT
    C(Z);:NEXT:PRINT
450 PRINT"[7*CRSR abajo]";:
    IF AA=1 THEN PRINT"-";
460 IF AA=2 THEN PRINT"+";
470 PRINT"[4*ESPACIO]
    [4*CRSR izq.]"A:PRINT
    "[CRSR arriba]"V$;
480 FOR Z=1 TO 8:PRINT D(Z);
    :NEXT Z:ZA=INT(SS/16):
    ZB=SS-(16*ZA)
490 PRINT"[5*ESPACIO]"MID$
    (Z$,ZA+1,1);MID$(Z$,ZB+1
    ,1):GOTO 150
500 I$="":PRINT"[FLECHA izq.]
    [CLR/HOME][21*CRSR abajo]
    NUMERO(ENTRE-128 Y +127)?
    [4*CRSR izq.]";
510 FOR Z=1 TO 4
520 GET J$:IF Z=1 AND (J$="-"
    OR J$="+") THEN U$=J$:
    PRINT U$;:NEXT Z
530 IF Z=1 THEN 520
540 PRINT"*[*CRSR izq.][]
    [CRSR izq.]";:IF J$=""
    THEN 520
550 IF J$=CHR$(13) THEN 610
560 IF J$=CHR$(20) THEN 500
570 IF ASC(J$)<48 OR ASC(J$)
    >57 THEN 520
580 I$=I$+J$:PRINT J$;:
    NEXT Z
590 GET J$:IF J$=CHR$(20)
    THEN 500
600 IF J$<>CHR$(13) THEN 590
610 IF VAL(I$)<0 OR (VAL(I$)
    >128 AND U$="-")OR(VAL
    (I$)>127 AND U$="+")THEN
    500
620 IF U$="-" THEN AA=2:GOTO
    640
630 AA=1
640 PRINT:PRINT"[CRSR arriba]
    [39*ESPACIO]";:A=VAL(I$)
    :GOTO 280
```

LA CONVENCION DEL SIGNO

Como hemos dicho, para muchos fines un número en binario o en hexadecimal corresponde perfectamente a un número positivo o negativo. Sin embargo hay ocasiones en que se quiere conocer si un número es positivo o negativo.

Los saltos realizados en los programas en lenguaje máquina requieren que se especifique el número de bytes que abarcan, en valores positivos si el salto es hacia adelante y negativos si el salto es hacia atrás. El ordenador examina el primer bit del número binario y determina por sí solo si se trata de un número positivo o negativo. Si el primer bit es un 1, el ordenador considera el número como negativo, mientras que si es 0 lo toma como positivo.

Este procedimiento se ajusta a una determinada convención de signos. Esto significa que el ordenador, en vez de manejar números binarios que varían entre 0 y 255, considera que su margen de variación va desde −128 a +127. En el programa de conversión en complemento a 2, habrás oído la señal de «bip» al llegar a 128. Esto se debe a que 128, que en binario es 1000 0000 y en hexadecimal es $80, corresponde a un valor negativo. (Haz la prueba: −1000 0000 + 1000 0000 = (1)0000 0000, que es 0 en binario de ocho bits; 80 + 80 = (1)00, que es 0 en hexadecimal de dos cifras; 128 − 128 = 0 en decimal.)

¿Qué sucede entonces? Dado que 1000 0000 tiene un 1 en su primer bit, el ordenador lo considera como un número negativo, el −128. Por otra parte el cero, que es 0000 0000, tiene un 0 en su primer bit, por lo que el ordenador lo toma como un número positivo.

SINTAX ERROR

En el número 12 de INPUT COMMODORE en el artículo «Aprovecha las interrupciones», pág. 45, aparecieron unas erratas en la línea 30. Esta debe quedar como sigue:

```
30 T$="000000":FOR Z=0 TO 5:POKE 837-Z,VAL(MID$(T$
,Z+1,1)):NEXT Z
```

Del mismo modo la línea 160 del listado de la página 46 debe quedar como se indica a continuación:

```
160 DATA 157,14,30,169,0,157,14,150,232,136,208,239
,76,191,234
```

En el artículo «Sólidos de revolución» hay algunos errores en las líneas 60, 130 y 1.510. Deberán quedar así:

```
60 LINE BX,BY,X,Y,2
130 IF CP=12 AND X-M>140 THEN X=X-M
1510 D=C:PY=IS*(Y*.5)
```



LOS MEJORES DE INPUT COMMODORE

PUESTO	TITULO	PORCENTAJE
1.°	Commando	19,2 %
2.°	Green Beret	18,7 %
3.°	Skyfox	13,9 %
4.°	Uridium	12,8 %
5.°	Rambo	11,7 %
6.°	Spindizzy	5,9 %
7.°	Hardball	5,8 %
8.°	Saboteur	4,6 %
9.°	Psi 5 Trading co	4,2 %
10.°	Biggles	3,2 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Noviembre de 1986.

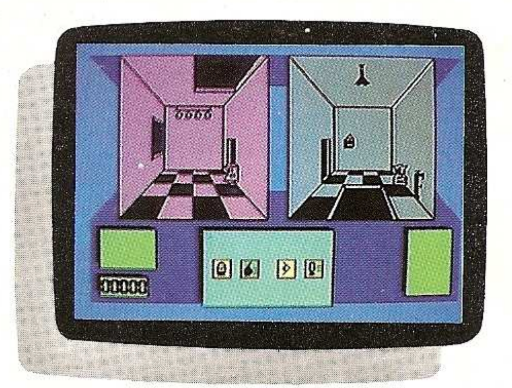


LOS DESACTIVADORES

Te encuentras al frente de un equipo de androides especializados en la desactivación de explosivos. Corre el año 2018.
La misión consiste en recoger las bombas que un grupo terrorista ha ido diseminando por las instalaciones de un importante centro secreto de desarrollo, el Instituto de Investigaciones Gravitacionales. Debes localizar los artefactos depositados en cinco edificios.
Existen dos inconvenientes. Por un lado los robots guardianes han sido reprogramados para atacar a todo lo que vean, sin preguntar. Por otro cada estancia tieneun diferente nivel de gravedad.
Lo más sorprendente de este juego de laberinto es el impresionante efecto tridimensional conseguido. Las habitaciones aparecen dibujadas en perspectiva y los androides ven afectado su tamaño en función de la distancia. Simultáneamente, aparecen visualizadas dos estancias adyacentes.
En la parte inferior de la pantalla verás un diagrama del edificio con la ubicación de los explosivos, así como la puntuación y la cantidad de ayudantes que te quedan. Recuerda, no hay puertas traseras, sólamente podrás salir por la principal.
Para complicar mas aún las cosas, deberás colocar determinados circuitos electrónicos en la sala del ordenador, sino surgirán graves problemas.
Mediante el joystick, eliges desde el mapa al andoride que deseas manejar en cada momento. Después podrás pasearlo por las habitaciones. Igualmente es factible examinar las habitaciones y soltar las bombas y los circuitos empleando la ventana de control de iconos (figuras representativas de los elementos). Existe una limitación, cada androide solamente puede llevar un elemento simultáneamente, por lo que debes ser hábil en las secuencias de manejo de bombas y circuitos, decidiendo si quieres pasar tu bomba a otro robot.
Aunque en un primer momento no lo pueda parecer, **Deactivators** es un juego de inteligencia, que exige una atención extremada unida a la necesidad de dotes de observación. Desde luego el entretenimiento está asegurado, aunque no se cacen marcianitos.







DATOS GENERALES

TITULO Deactivators

FABRICANTE Tigress Marketing

ORDENADOR Commodore 64

TEMA DEL PROGRAMA Desactivador de bombas

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	9
GRAFICOS	9,5
COLOR	7
SONIDO	6
TOTAL	40,5

★★★★★★★★★★★★★★★★★★★★★★★★★★

MAIL ORDER MONSTERS

Mail Order Monsters es un curioso título para un juego muy curioso, que te llevará a tierras y tiempos muy lejanos, perdidos en la prehistoria de algún planeta desconocido.
El programa posee tres niveles de juego. El primero es una especie de presentación reducida de lo que sucederá en los dos siguientes.
Todo se compra y se vende, puedes pedir a la fábrica de "morfos" el monstruo de tu preferencia, partiendo desu catálogo de doce criaturas.
Lo mas excitante del juego es el torneo. Solo frente al ordenador o contra un amiguete, deberás elegir al monstruo que te represente. Los monstruos lucharán en distintos sectores, desplazandose por ellos a golpe de joystick. Con el propio botón de disparo obligarás a tu servidor al ataque en su especialidad; algunos llegan incluso a escupir. Pero ojo, debes vigilar sus variables







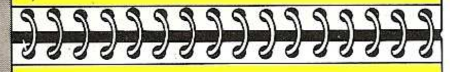
DATOS GENERALES

TITULO Mail Order Monsters

FABRICANTE Electronics Arts

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Mezcla entre Arcade y Juego de Estrategia



CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9,5
INTERES	10
GRAFICOS	9,5
COLOR	9
SONIDO	8
TOTAL	46

vitales para conocer en que momento físico se encuentra.
Existen otros monstruos desperdigados por este mundo tenebroso. Pueden aparecer en cualquier lugar y atacarán a cualquiera de los contricantes que les apetezca.
El juego tiene multitud de recursos que complican tu acceso a la victoria y que decubrirás cuando lo practiques.
Tu principal baza consiste en elegir un monstruo capacitado, pero en ellos nada es redefinible. Por ejemplo, existe una abeja venenosa, un encantador de brontosaurios, etc.. Busca bien entre los doce disponibles, que podrá ser un monstruo baboso, o quizás tenga una armadura inalterable o una inteligencia digna de envidia. También varia la fuerza física, el llamado cociente vital etc...
Es en su segunda y tercera fases donde este programa se distingue de los demás. Para comenzar, te encuentras en medio del llamado "Centro de selección". Por medio del joystick puedes desplazarte hacia el llamado "Centro de compras", que no es sino un supermercado del horror. Los quinientos Psychons (es el nombre de la moneda de curso legal en ese planeta) te servirán para comprar el monstruo de tus sueños. Lee atentamente las ventajas e inconvenientes de cada uno y, sobre todo, trata de evaluar su relación calidad/precio. Un monstruo baboso por ciento ochenta Psychons puede ser un muy buen negócio, si tras esto compramos un poco más de cerebro para él. Un poco mas de fuerza y coraje no le vendrá nada mal tampoco. Pero cuidado, no gastes todo lo que tienes, porque queda un apartado más: el armamento. Este

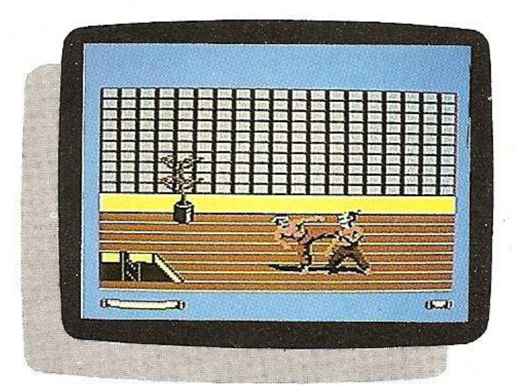
cubre primero sus necesidades de defensa. Podrás elegir desde un laser de protones defensivos hasta un tentáculo extra, o simplemente una armadura "de las baratitas". Salimos del almacen. Ya tienes un magnifico animal pero aún no es lo bastante peligroso teniendo en cuenta el barrio. Habrá que darse un garbeo por la armeria para selecionar algo conque defenderse nuestro pobre bicho, tal vez una espada, un lanzallamas, una granada e incluso el pedrusco "vulgaris". Aquí la seleción del armamento ha de ser acorde con las estrategias de lucha que pienses adoptar. El almacén dispone de más de treinta armas diferentes, cada una tiene una utilidad muy definida. Por lo demás la acción se desarolla como en la primera fase.
Idem para la tercera fase, que requiere un serio entrenamiento con respecto a las dos anteriores. Cada "morfo" será asociado a cada tipo de lucha, a cada circunstancia y a cada tipo de lucha. En esta última etapa todo es redefinible: alimento del animalito, botiquín, vitaminas, etc...
A pesar de una animación gráfica que no sorprende, este juego dispone de una serie de opciones que enriquecen considerablemente la acción del juego, sobre todo en lo que se refiere a la estrategia. No hay que olvidar que éste es un juego de estrategia con gran dosis de interactividad.
Por otro lado, estamos ante un programa totalmente adaptado al castellano para mayor integración del usuario en las aventuras.

****

CONTINUACION DE EXPLODING FIST

Exploding Fist II: la leyenda continúa Tras unos segundos de espera, verás la imagen de un karateka pegando una patada contra una mesa como anticipo de lo que viene a continuación. Eres un activo luchador que debe luchar hasta el fin con un señor de la guerra. El malvado tiene su guarida en un volcán.
Pasas por una etapa previa de aprendizaje de nuevas técnicas marciales. La recolección de objetos dispersos por las pantallas y lograr victorias ante quince oponentes aportan esos nuevos conocimientos.
Exploding Fist II es un juego similar al anterior en cuanto a surtido de patadas: voltereta atrás y adelante, golpe de mano a la cara, el estomago y las piernas, etc.
Las técnicas permitidas son idénticas, pero la calidad gráfica difiere mucho, ya que su predecesor estaba planteado sobre pantallas estáticas. Ahora se dispone de un fondo continuo que se desplaza en scrolling.





DATOS GENERALES

TITULO Exploding Fist II

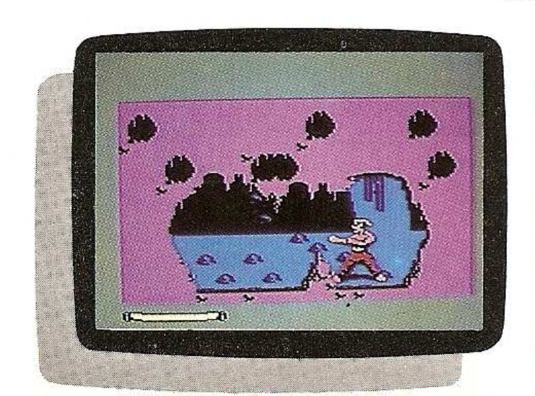
FABRICANTE Melbourne House

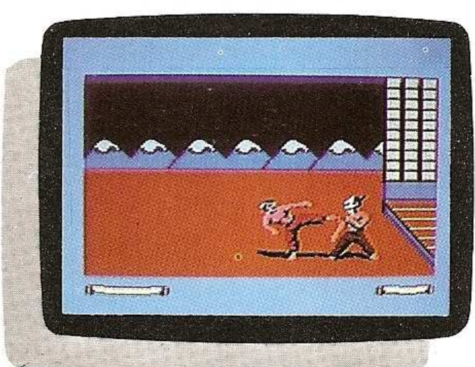
ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Artes Marciales

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7,5
INTERES	8
GRAFICOS	8
COLOR	6,5
SONIDO	6
TOTAL	36





El karateka se desplaza por una mansiòn china en un escenario tropical.
Apareces sobre el tronco de un àrbol situado sobre un rio con abundante de vegetaciòn. Pasas a otro tronco, pero en este punto eres interceptado por un atacante enmascarado, que te propicia una magnifica patada en toda la cara.
La experiencia con el joystick que adquiriste en la primera parte serà de vital importancia para salir victorioso en la nueva aventura.
La masiòn perece un rascacielos mas que una casa china, porque nunca se acaban las plantas. Cada planta tiene un oponente de mayor nivel que el anterior, haciendose mas lenta la progresiòn hacia el final.
El juego acaba cuando logras vencer al debilitado y perverso señor de la guerra, tras eludir sus trampas y derrotar a sus guardaespaldas en la cueva del volcàn.
Es un juego clàsico de artes marciales con elogiables caracteristicas gràficas y elevado nivel de realismo en los movimientos.

★★★★★★★★★★★★★★★★★★★★★★★★★★

PARALLAX

La saga **Uridium** ataca de nuevo. Este juego es otra de sus variantes, que esta vez permite el scrolling (desplzamiento) en las cuatro direcciones.
Debes escapar de los marcianos que se supone eran amigos, una vez que descubres algo que no debieras conocer: sus planes para invadir nuestro planeta. Tienes que lograr que los planes de invasiòn sean conocidos por los confiados terrestres.
Antes de marcharte tendràs que recoger a los cientìficos que colaboraban en el planeta ahora enemigo.
Aunque tu zona de vuelo aparece a vista de pàjaro, puedes variar tu altitud pasando por debajo o encima de los obstaculos que se interponen en tu ruta. La sombra que proyecta sirve a su vez para que evalues la altitud de la nave.
Existen varios niveles y pasas de uno a otro una vez que franqueas la barrera de





DATOS GENERALES

TITULO Parallax

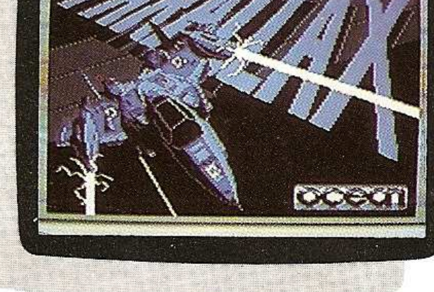
FABRICANTE Ocean

ORDENADOR Commodore 64

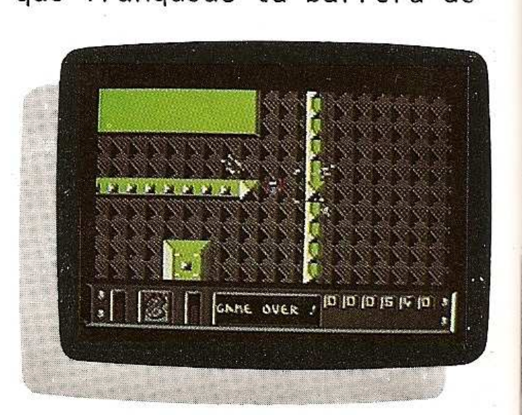
TEMA DEL PROGRAMA Combate espacial

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	8
GRAFICOS	9
COLOR	8
SONIDO	7
TOTAL	40







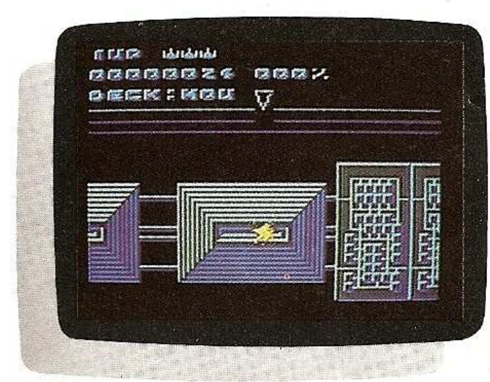
seguridad. Deberás conocer un código que permite la destrucción de la linea que impide tu avance. Permanece atento a los mensajes que aparecen por una ventana al pie del mapa.
Es de destacar la limpieza del scrolling del fondo. La música es también otro elemento que demuestra lo elaborado del programa.

★★★★★★★★★★★★★★★★★★★★★★★★★

WAR

La primera impresión que nos produce este nuevo juego es que se trata de una versión ampliada del clásico **Uridium**
Aquí tomas el protagonismo en forma de un habilidoso piloto que maneja la nave de guerra mas avanzada tecnologicamente que existe y debes librar a la Tierra de los molestos invasores.







DATOS GENERALES	
TITULO	War
FABRICANTE	Martech
ORDENADOR	Commodore 64
TEMA DEL PROGRAMA	Combate Espacial

CALIFICACION (Sobre 10 ptos.)	
ORIGINALIDAD	6
INTERES	8
GRAFICOS	8
COLOR	8
SONIDO	9,5
TOTAL	39,5

Nuevamente eres el elegido. Se cierne sobre el planeta la amenaza marciana, que avanza con la pretensión de adueñarse del planeta azul.
La conformación extraterrestre está formada por veinte cilindros unidos entre sí, posibilitando la evacuación urgente en caso de ser dañado cualquiera de ellos. Cada uno tiene su cometido, sea miltitar, residencial, etc.
Tendrás que ir destruyendo a los invasores y despues entrar en cada cilindro para terminar con lo que queda de él, pero el tiempo es limitado.
Despues habrás de elegir un código de colores correcto para lograr salir.
a fuerza centrípeta, producida mediante el giro de los cilindros, actua como gravedad hacia ellos.
Tienes que burlar y deshacerte de las patrullas de control que trataran de cazarte.
El manejo de la nave se efectua, como en habitual, con el joystick. Una vez que penetras en el interior de un cilindro, comienza automáticamente la cuenta atrás para su autodestrucción. Debes llegar al panel que contiene cinco filas de celulas de memoria y cursores de movimiento en X e Y. El color de la cerradura se verá en las partes superior e inferior de la pantalla. Tienes que apuntar el cursor a la celdilla del color marcado para abrir la puerta. La rapidez es fundamental.

★★★★★★★★★★★★★★★★★★★★★★★★★

ASTERIX Y EL CALDERO MAGICO

Los simpáticos amigos **Asterix**, **Obelix** y **Panoramix**, vuelven a las andadas en tierras de la Bretaña francesa.
Por si no lo recordais, **Asterix** es el pequeño y mas cordial de la pandilla.También es listo,

pero no es tan fuerte como su amigo el regordete **Obelix** porque el no cayó en un caldero de brebaje mágico cuando era un bebe. Desde entonces **Obelix** es fuerte como un mulo y le está prohibido beber nuevamente del mismo. Sin embargo, siempre trata de ver si puede conseguir un poco.
En uno de esos intentos, es cuando cobra vida el juego. **Obelix**, el repartidor oficial de menhires, hace cola para recibir su ración de brebaje. El druida **Panoramix** le dice que "verdes las han segado..".
En un momento de enfado, **Obelix** le pega una fuerte patada al caldero, rompiendolo en 8 trozos ante los ojos atónitos de los miembros del pueblo. **PanoramiX**, sin perder un momento, recoge el poco líquido que no ha absorvido la tierra y se lo entrega a **Asterix**.
Nuestro héroe marcha en busca de los pedazos dispersos del caldero, ya que es imprescindible para preparar la pócima, y sin ella no podrán vencer nunca a los romanos ".
Esta es la ingrata misión de nuestros personajes.
En la pantalla aparecen varios indicadores. El primero es el número de jamones que tienen los amigos **Asterix** y **Obelix**. Comprobareis que no pueden llevar mas de 5 a la vez, pero empiezan con solo dos. Así que es recomendable darse una vuelta por el bosquecillo situado al este del campamento.
También se ve un dibujo que representa al caldero, el número que aparece a su lado indica la cantidad de piezas recogidas.
Otros tres indicadores muestran las vidas disponibles, si tienes una llave muy útil para abrir cualquier puerta que encuentres y, por último, el consumo del brebaje.
Cuando quieras mas fuerza, basta con que dispares rápidamente para beber unos sorbitos de la fabulosa pócima .
El juego está bien resuelto. Los gráficos se generan en la pantalla y tardan dos o tres segundos en realizarse. El sprite de los dos principales personajes no están en la linea de este fabricante. Un curioso sistema de ventanas es utilizado para las escenas de lucha.
Cuando **Asterix** se encuentra con un enemigo, o un jabalí, aparece una ventana sin color en la que se ve ampliada la escena.
Existirán muchas ocasiones de pelea, porque las piezas del caldero se hallan siempre en territorio enemigo y abundan las patrullas de esos locos de romanos.





DATOS GENERALES

TITULO Asterix y El Caldero Mágico

FABRICANTE Melbourne House

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Aventuras de Asterix

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	8
GRAFICOS	8
COLOR	8,5
SONIDO	6
TOTAL	39,5

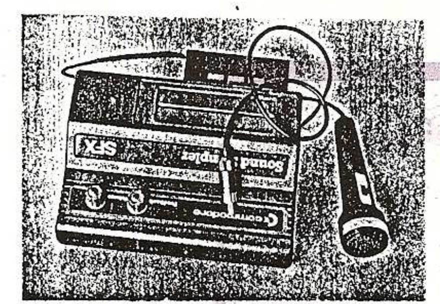




COMPULAND
Calvo Asensio N.8
Madrid 28015

CARTUCHOS

FINAL CARTRIDGE ultima version con Freezer	9.900
FREEZE FRAME para copias de seguridad pasa C a C,C a D,D a D y D a C.con copiador para programas secuenciales,formateador rapido(10 sg),y los comandos del dos en modo directo	11.900
ISEPIC Pasa de cinta a disco en pocos bloques,permitiendote entrar dentro del programa desprotejido	11.900
SECURITY CASS Interface copiador de cinta a cinta	4.900
SOUND SAMPLER Muestreador analogico digital.Permite almacenar para su posterior manipulacion cualquier sonido real,voz, instrumentos(via microfono o linea).Con posibilidades varias como tonacidad,eco,reverse,retardo,conexion MIDI,.....,..	27.500
SOUND EXPANDER Amplia a ocho voces las posibilidades del sintetizador.Sintesis de sonidos en FM.Incorpora Patrones Ramios y melodicos asi como bajos.Posibilidad de componer a tiempo real o sobre partitura , enlace MIDI	32.500
SUPER GRAPHIX Da infinitas posibilidades a tu impresora	21.950
LASER Turbo acelerador de disco con comandos del dos	4.950
QUICK DISK PLUS + Turbo acelerador comandos del DOS	4.950
DIGIDRUM Convierte tu Commodore en una bateria	13.900
GRABADOR DE EPROM	LLAMAR
SUPER DOLPHIN Turbo acelerador entre 20 y 25 veces con monitot de codigo maquina,programa las teclas de funcion y muchas otras opciones	16.000
SIMONS BASIC Da 114 comandos mas a tu Commodore	14.500
EXPANSOR DE CARTUCHOS conecta varios cartuchos a la vez	5.900
INTERFACE MIDI	LLAMAR
VENTILADOR AXIAL	LLAMAR
FUENTE ALIMENTACION 64	6.500





HARDWARE

128 D	130000
64	40500
64c	44500
1541	49900
1541C	49900
NL10star	83000
Seikosa	57500
Riteman	67.500
Monitor F	28000
Monitor C	65000

UTILIDADES

Disponible todo de Casa de Software
Cimex
SEINFO
HISPASOFT
FERRE MORET
SAKATY
y productos impor tados por nosotro

NOVEDADES

FIST II - GHOST'N GOBLINS - PARALLAX - SOUTHERN BELL - SILENT SERVICE
SUPER CYCLE - MAIL ORDER MONSTER - ALLEY CAT - WAR - TRAP - HACKER II
ASTERIX - DRAGONS LAIR - MISION A.D. - KNIGHT RIDER - SOUTHERN BELL
INFILTRATOR - BIGLESS - EXPRESS RAIDER - GREAT ESCAPE - MIAMI VICE
SPITFIRE 40 - FLIGHT DECK - TIME TRAX - EQUINOX - WAR PLAY - ACE
TUBULAR BELL - SHOGUN - ULTRA GAMES - FIONA - MOVIE - BATMAN - GUNSHIP
MARBLE MADNESS - TAU CETY - MERCENARY - SPINDIZY - PAPER BOY - MICKY
y cerca de mil titulos mas que continuamente van aumentando.

EDUCATIVOS

El estudiar no tiene por que ser pesado y una incordia , puede ser divertido y una competicion entre varios amigos,teniendo al ordenador como arbrito.
Hemos lanzado al mercado una serie de programas EDUCATIVOS para 5,6,7,y 8 de EGB , hecho por profesores de acuerdo a las normas del MINISTERIO DE EDUCACION Y CIENCIA y que resumen las editoriales mas importantes de esta curso.
Dele la oportunidad a su hijo de que pueda competir contra el ordenador o sus propios companeros , en un programa que poco a poco le ayudara a dominar los temas que esta estudiando en este curso.
Si desea recibir mas informacion escribanos o llamenos por telefono.
CATALOGO GRATUITO
Si quieres recibir periodicamente informacion sobre ultimas novedades y articulos a la venta fotocopie o recorte este cupon y envielo.
NOMBRE...
DIRECCION..
POBLACION................................D.P.....................
TELEFONO......................... ORDENADOR......................
CALVO ASENSIO N.8 MADRID 28015 TFNO 2431638 COMPULAND

UNA SESION DE GOLF

DATOS GENERALES

TITULO Hole in One

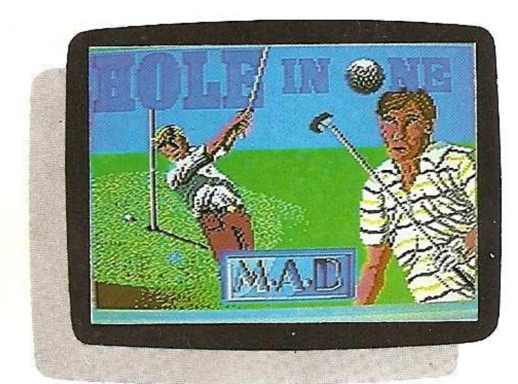
FABRICANTE Mastertronic

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Juego de Golf

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	7,5
GRAFICOS	9,5
COLOR	9,5
SONIDO	7
TOTAL	41,5



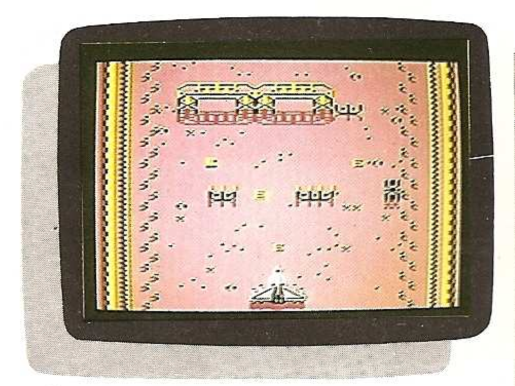


El sano deporte del golf ha marcado una tendencia en los videojuegos. **Hole in One** es otra versión ue viene a enriquecer el surtido de programas ya existentes.
Hay dieciocho hoyos y puedes elegir el tipo de palo a utilizar en cada golpe de joystick.
Dispones de la información precisa: distancia y par, así como la velocidad y dirección del viento. En la carátula del programa aparece también el mapa del campo de juego.
Los gráficos muestran una perspectiva tridimensional. Puedes jugar con el teclado, pero es preferible el joystick, aunque en la muchos casos el reflejo del movimiento no pareció ser el que esperabamos.

★★★★★★★★★★★★★★★★★★★★★★★★★★★

ALLEYKAT

Nuevamente el **Uridium** parece haber sido la pieza inspiradora de otro juego. **Alleykat** es una carrera por el máximo título. Esta vez el movimiento de la nave y el fondo se produce en sentido vertical, desde abajo hacia la parte superior de la pantalla.
Existen obstaculos, tales como arcos, que debes sortear para no estrellar la nave.
Se supone que la zona de vuelo son ruedas situadas en el espacio y existen ocho diferentes, con colorido e incluso





DATOS GENERALES

TITULO Alleykat

FABRICANTE Hewson

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Combate Espacial

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	7
GRAFICOS	9
COLOR	9
SONIDO	7
TOTAL	39

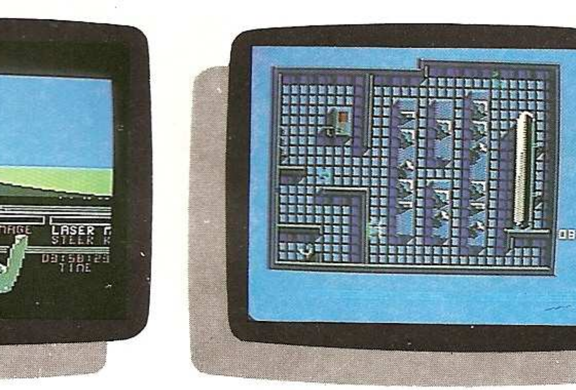
superficies distintos en cada caso. Para ganar el premio en una de las treinta y dos carreras, deberás ser cuidadoso al elegir. Lo harás de acuerdo con el dinero disponible y tus habilidades particulares.
Las carreras incluyen ocho tipos de prueba, que son una ligera variante entre sí. Se puede optar por volar a toda marcha, complicandose las posibilidades de supervivencia. Se puede realizar vuelo de fondo, procurando únicamente permanecer intacto.
Destruir el paisaje, sin dejar piedra sobre piedra, es otra opción. Incluso puedes practicar el slalom entre los obstaculos de la pista.
Comprueba el número de vueltas que das, dejandolo cuando estés contento con tu puntuación, para después acudir a la siguiente carrera.
El tiempo se mide en extraños meses galácticos, realizandose una convocatoria por mes
En la superficie aparecen depositados caracteres con la forma de la letra E, que son puntos de energía. De acuerdo con tu habilidad, te será permitido llevar mas de los cinco iniciales energéticas.
Los efectos de sonido han sido cuidadosamente programados, lo mismo se puede decir de la música.

****

EL COCHE FANTASMA

Este coche fantástico parece haber salido con cierta premura, sin duda provocada por la ostentosa campaña publicitaria en las revistas de Gran Bretaña. Por otra parte se había creado una gran espectación en torno a este título.
Has asumido la personalidad de **Michael Knight** y viajas a bordo del increible K.I.T.T.
Al comenzar el juego te encuentras en la cuidad de Atlanta (Georgia, E.E.U.U) y recibes una comunicación de **Devon** en la que este te pone en la pista de un grupo terrorista que está tratando de perturbar la paz mundial, pretendiendo asesinar al embajador ruso. La verdad es que se trata de una tarea muy difícil la encomendada a nuestro héroe del momento, ya que a lo largo del juego -tal y como sucedía en la realidad- no dispone de armas.
El juego se desarolla en base a tres pantallas diferentes. Una muestra el mapa de los Estados Unidos, en el que algunas cuidades aparecen reseñadas (Miami, Chicago, Dallas...).
Verás que en esta pantalla la cuidad en que te encuentras parpadea y la primera indicación que aparece bajo el mapa te dirá lo que tienen los





DATOS GENERALES

TITULO Knight Rider

FABRICANTE Ocean

ORDENADOR Commodore 64

TEMA DEL PROGRAMA
Serie de Televisión

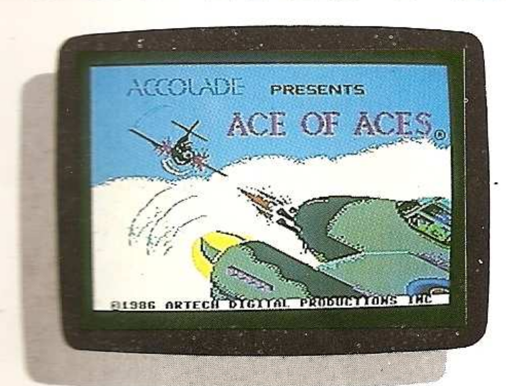
CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8,5
INTERES	8
GRAFICOS	9
COLOR	8
SONIDO	8
TOTAL	41,5

terroristas en esa cuidad (su armeria, sede o bien el blanco).
La segunda pantalla de juego es la que mas a menudo aparece en el juego.
Es una vista en perspectiva de la carretera en la que serás atacado por una serie de helicopteros.
En esta etapa el juego permite dos opciones: **KITT** conduce y tu disparas o al revés.
Del supersalpicadero de **KITT** que aparece en la carátula del juego solo quedan unos pocos controles: un indicador digital de velocidad, un contador de tiempo, un indicador de daños, y un indicador de calentamiento del laser. No queda nada del turbo, etc...
El efecto de conducción al girar el volante no es demasido rico y ni siquiera se mueve el que se ve en la pantalla. Tampoco aparecen otros vehiculos.
Una interesate posibilidad consiste en la detención del juego, presionado la tecla RUN/STOP.

★★★★★★★★★★★★★★★★★★★★★★★★★

SOFTACTUALIDAD

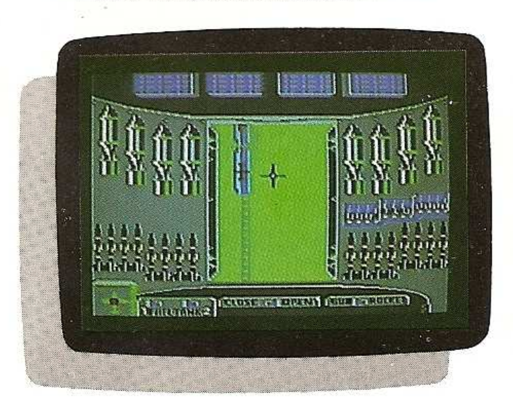




AS DE ASES

La harto conocida firma **Accolade**, popular por el extremo cuidado con que elabora los gráficos de sus programas, ha adaptado un Best Seller europeo:**Dam Busters.**
La verdad, no es solamente una adaptación, sino una enorme mejora respect al programa original.
Existe ina opción de entrenamiento, pasando después al de ataque real y al ataque a una presa, sea convoy de treneso incluso a grupos de submarinos. La animación es extraordinaria y el sonido esta totalmente en la linea de la casa.





★★★★★★★★★★★★★★★★★★★★★★★★★

HACKER (SEGUNDA PARTE)

Para todos los que se volvieron locos con **Hacker I, Activision** lanzó la segunda parte de este fenomenal juego de estratégia y acción.
Esta vez la trama se desarrolla en la central de inteligencia de los sovieticos.
En cuatro ventanas verás simultaneamente el desarrollo del juego.
Deberás utilizar cinco robots para sonsacar a los científicos rusos y evitar así que lleguen a crear el arma mortífera que pudiera romper el inestable equilibrio mundial.
Esta vez los robots van equipados de cámara, acceleradores, monitores, busca personas, etc...
En cualquier caso, hará la delicia de todos los que disfrutamos de la primera parte.

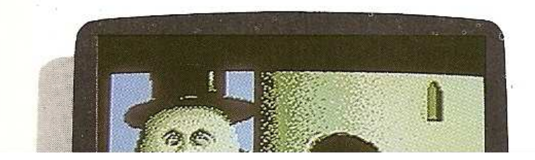


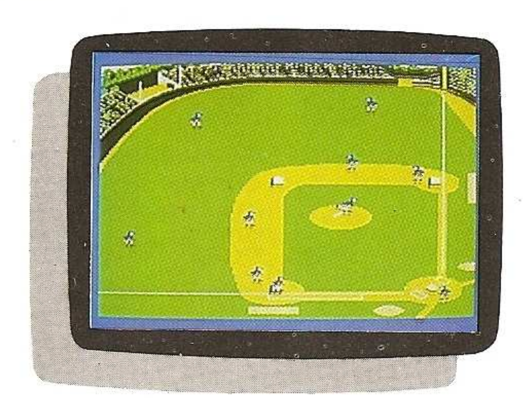
★★★★★★★★★★

THE PAWN

"El juego conversacional **The pawn** es el mejor juego de aventura que jamás se ha

relizado en cualquier sistema, aunque únicamente existe por ahora en versión **ATARI...".**
Esto era cierto hasta hace muy poco: ya que la versión para **Commodore** pronto saldrá al mercado y hay que ver la calidad que alcanzan estos programadores de **Rainbird,** incluso el planteamiento del juego es interesante.
Cada pantalla aparecerá partiendo de un scrooling desde la parte superior,





resaltando particularmente por la calidad gráfica, si lo hace por la variedad de la animación de los jugadores.
No hay que perderse el efecto del jugador que se tira en plancha para alcanzar una de las

algo novedoso, sin embargo **Chess Mate 2000** lo consigue plenamente.
Imaginaros un juego de ajedrez en el que puedes regular el número de jugadas de previsión que realizará el ordenador.
Este es un juego que te enseña a ganar indicandote todos los movimientos que puedes realizar con las piezas, que tiene opción bi y tridimensional y muchas cosas más. Con el que puedes pedir ayuda en los momentos difíciles y que



ERROR: ioerror
OFFENDING COMMAND: image

STACK: